# Chapter 5
# CS Perspectives and TEL

**Abstract** This chapter analyzes the relationships between CS and TEL: the different roles that CS may play within the TEL field, and the different ways in which computer scientists may conceptualize their involvement.

An[1] intrinsic difficulty in clarifying the relationships between TEL and CS is that the CS field encompasses a number of distinct intellectual traditions, leading to debates about the nature of this field. It can be conceptualized and broken down into different sub-domains in different ways, and is subject to different definitions, interpretations, and misunderstandings.

While we shall not attempt here a tentative definition of CS or a description of its history and different features, it is worth highlighting that what the term CS emphasizes may correspond to different notions, in particular:

- Emphasis on the studies of computation, complexity and formalisms – what is often termed "theoretical CS", and takes its origins from the mathematics tradition.
- Emphasis on hardware and software development, software applications for specific domains, software engineering, etc.

As a matter of fact, very little TEL research or practice may be considered as related to the theoretical foundations of CS. TEL is rather concerned with software development, and may be powerfully addressed as an engineering field (see Chap. 6). In TEL, the human and social aspects of computer system design, usage and evaluation are central issues.

Considering the relationships between CS and TEL is useful for both computer scientists and educationalists as it helps to clarify and understand the nature of the

[1]Part of this chapter is inspired by: Tchounikine, P., Mørch, A.I. & Bannon L.J. (2009). A computer science perspective on Technology Enhanced Learning research. In: Balacheff, N., Ludvigsen, S., de Jong, T., Lazonder, A. & Barnes, S. (Eds.) Technology-Enhanced Learning – Principles and Products, Berlin: Springer, 275–288.

P. Tchounikine, *Computer Science and Educational Software Design*,
DOI 10.1007/978-3-642-20003-8_5, 

work and the dimensions underlying actors' engagement. In order to investigate these relationships, we explore two different perspectives. First, in Sect. 1, we take a TEL perspective and disentangle different roles for CS in TEL: (1) building new technologies or further developing existing technologies to create novel possibilities for supporting human activities; (2) elaborating powerful abstractions; (3) enabling specified models and processes to be run on computers. Second, in Sect. 2, we disentangle three prototypical ways in which computer scientists may conceptualize their involvement in TEL: (1) TEL as a place for clever CS applications, (2) TEL as a field where some CS problems arise and are to be solved and (3) TEL as a proper field. While the first perspective emphasizes a characterization of different types of work, the second emphasizes the different views on TEL that may underlie the work. This analytical analysis is completed, in Chap. 6, by an analysis of why and how developing an engineering perspective to TEL is a productive approach.

## 1 Roles of Computer Science in TEL

To clarify the different ways in which CS impacts the field, three prototypical (non-orthogonal) roles for CS may be disentangled:

1. Building new technologies or further developing existing technologies to create novel possibilities for supporting human activities.
2. Elaborating powerful abstractions.
3. Enabling specified models and processes to be run on computers (algorithm development and implementation).

These roles are not orthogonal to each other and, in the context of a given project, computer scientists may act at different levels and on different planes. Their separation, however, provides a basis for clarification of the nature of computer scientists' actions (see Sect. 3).

### *1.1 Creating Novel Possibilities for Supporting Human Activities*

As highlighted in the introduction, CS is a broad field that includes work on the creation of new computer-based technologies (i.e., hardware, software or user interfaces). Computer scientists are naturally at the forefront of this activity, which has a spin-off effect in that it opens up new possibilities, that is, creating software or hardware allowing innovative interactions to take place.

The evolution of the TEL field and of the considered types of systems can easily be linked to technological advances in CS:

- Behavioral-inspired systems such as basic drill and practice systems, involving the direct application of algorithmic techniques.

- Intelligent Tutoring Systems based on representation of knowledge and expertise, decision making, Artificial Intelligence and knowledge engineering techniques to model domain expertise or pedagogical expertise.
- Microworlds or simulations based on modeling and visualization (2D, 3D) techniques.
- Hypertext and multimedia based on structuring documents and creating conditional or dynamic access techniques.
- Communication-based systems and CSCL, based on the technological developments in networked computing (basic exchange of data between two computers, speed rate improvements, graphic interfaces, permission or deadlock management allowing the creation of advanced systems such as wikis or workflows) and Human-Computer Interface (HCI) techniques.
- To give other examples: Web-based learning approaches; mobile learning; so-called e-learning 2.0 or Grid-learning; etc.

This list is witness to the impact of the evolution of technology: new types of educational systems not only become possible, but are imagined because some technology, that did not exist previously, creates new possibilities.

It may also be noticed that advancement of technology impacts not only the very nature of systems, but also how some dimensions of already existing types of software are thought of. For instance, the advancement of data mining and text mining in CS has raised interest for interaction analysis. Recent work in CSCL attempts to provide learners with adaptive frameworks and individual or collective support using ITS techniques[2]: the dimensions of collaborating for learning and the intrinsic nature of the system are not impacted, but are enhanced by a new dimension.

Of course, the evolution of technologies is only part of the story of the evolution of the field, see the discussion related to how the CSCL paradigm gained importance and the ITS paradigm became less dominant in Chap. 3.

## *1.2 Elaborating Powerful Abstractions*

### 1.2.1 Computer Science and Modeling

Modeling is a core issue in CS. Advanced software engineering practices such as the Model Driven Engineering[3] approach push forward the use of models to direct the course of understanding, design, construction, deployment, operation,

[2]Tchounikine, P., Rummel, N. & McLaren, B.M. (2010). Computer Supported Collaborative Learning and Intelligent Tutoring Systems. Nkambou, R., Bourdeau, J. & Mizoguchi, R. (Eds.): Advances in Intelligent Tutoring Systems, SCI 308, Berlin: Springer, pp. 447–463.

[3]See for example Model Driven Architecture (http://www.omg.org/mda/; retrieved November 11, 2010).

maintenance and modification of systems implementation. Working at the level of models rather than at a symbolic level facilitates software construction and, in particular, appears to be the only way to address the complexity of today's software and architectures.

Technically, it is now possible to generate a large part of the code required to build applications from abstract descriptions represented in conceptual modeling languages, which allows one to focus on abstract notions and processes rather than on programming language issues (e.g., variables or control structures). This is a step in the direction of software built by automated transformations of models. Software components or frameworks (i.e., high-level building blocks that hide their internal structures or code and expose their external interface and connection points) also facilitate and enhance software construction by addressing issues at a higher level of abstraction than general programming languages allow.

Models and components support the tendency of application developers to act at a level that is closer to that of the domain expert users. For instance, considering the design of software for a bank, some issues will be addressed at a purely technical level (e.g., networking issues), but some computer scientists will also be concerned, along with domain (bank) experts, to help model the notions to be considered (e.g., monetary transactions), and the processes to be operationalized.

### 1.2.2 TEL and Modeling

TEL is an area where building models that involve software issues is a key dimension, and elaborating (with learning scientists) powerful abstractions is arguably computer scientists' core contribution to TEL.

Modeling may take place at different levels such as CBPSs (which requires consideration of the role and the properties of the computer-based system in the setting) or modeling the notions required to design and implement software. For instance, software thought of as a *milieu* reacting to learners' actions requires consideration of a variety of models related to the taught domain knowledge, teaching scenario, learners' possible actions, learners' actions interpretation process, feedback model, etc. (see Chap. 8, Sect. 2).

Computer scientists are key actors in this type of modeling. They provide conceptual languages and notations (XML-based formalisms, UML-based formalisms, Petri nets, etc.), tools (design tools such as graphical modelers, visualization or simulation tools), and their modeling skills.

It should be noticed that such modeling activities have a value in and by themselves, and not only when used as a base for implementing procedures or processes. Very often, the objective of the work is to elaborate a model that allows one to understand or design the setting, even if no computer implementation is planned or the model is not implemented as such in the computer. For instance, it is very common to address learning scenarios via some given Educational Modeling Language and the associated conceptual notions or manipulation tools.

This modeling is often not made further use of, for example, to generate some code or tune a technical architecture, but it is still a valuable pedagogical tool.

A model is an oriented representation of something, which captures the features that have been decided to be pertinent given one's objectives. Modeling requires the addressed object to be understood, and what is pertinent to be decided. In the context of TEL, this is often not addressable by segmenting it into disciplinary phases, considering educational dimensions and then CS. As examples: the models underlying *Bio-sim*'s cognitive tools; the way problem solving and collaboration may be conceptualized and supported in *Colab-solver*; the way *JavIT* implements diagnosis or feedback; etc. Such situations require the problem to be conceptualized and addressed as a transdisciplinary TEL problem, in a way transcending the boundaries of conventional disciplines.

### *1.3 Implementing Specified Models and Processes on Computers*

Finally, the most obvious way for computer scientists to be involved in TEL is to be in charge of writing the code and implementing a working model on a computer system.

When considering this type of work, if the operationalization process raises some effective difficulties it requires the computer scientist to act as a CS researcher; if not, it requires the computer scientist to act as an engineer.

Implementation of educational software does raise effective CS research issues in some cases. As examples: creating knowledge-based systems that can implement pedagogical decisions; analyzing learners' interactions or outputs; interoperating software components on the fly; implementing tailorability; etc.

## 2 Engagement of Computer Scientists

In Sect. 1, we have disentangled different roles played by CS and computer scientists in TEL. In this section, we now consider the way in which computer scientists may conceptualize their involvement in TEL. We disentangle three prototypical cases:

1. Considering TEL as a place for clever CS applications.
2. Considering TEL as a field where some CS problems arise and are to be solved.
3. Considering TEL as a proper field.

The rationale for considering this dimension is that in order to understand what actors do, how they do it or the underlying rationale, it is also important to consider what their perspective is. Similarly to the CS roles introduced in Sect. 1, these perspectives are not orthogonal to each other. The purpose of disentangling and contrasting them is to highlight their features.

## 2.1 TEL as a Place for Clever CS Applications

Educational software may be approached as a place for clever applications of CS work. Within such an approach, emphasis is on innovation and creating novel possibilities. Typically, corresponding work builds on technical advances: a newly developed technology is interpreted as a means for creating pertinent novel possibilities for supporting human activities, and is turned into a new generation of educational systems or features.

This view of TEL is developed and enacted by a much wider audience than computer scientists. It corresponds to the technologically-pushed dimension of the field.

### 2.1.1 Negative Dimensions of Building on Technological Advances

Considering TEL as a place for clever applications of technological advances leads to technology-driven educational applications. The risk is of course techno-centrism and putting emphasis on the technology *per se* rather than on the pedagogical outcomes for learners. This may lead to the development of smart software that, unfortunately, does not correspond to any educational need or interesting usage.

Considering technology-driven educational applications, a recurrent pattern can be identified:

1. New technology stirs up a wave of excitement among technologically-oriented TEL researchers and early adopters in educational fields.
2. Disappointment appears as the expected pedagogical benefits do not immediately materialize.
3. The simple but core principle that technology is not the answer to a question that must be found, but rather a potential means to address well-defined educational problems, again resurfaces.
4. The importance of pedagogy over technology is once more stressed, and the technology-driven development stalls.
5. A novel technology occurs, followed by a new wave of educationally potentially useful computing innovation. Technologically-inspired visions come to the fore and take center stage for a period.

The fact that every new wave of techniques generates a new type of system, and often tends to replace the preceding one, is an issue that should be analyzed. Rather than deciding whether the new technologies allow one better to solve the educational problems that were already the subject of work with the previous technologies, or provide a better understanding of the educational objectives that should be addressed, the target (considered educational issues) seems to move with the means (the technology). When considering techno-pushed applications, the question to ask is: what is the specific educational advancement that is related to the fact that this particular technology is being used?

### 2.1.2 Positive Dimensions of Building on Technological Advances

Indeed, it is not because some new technology becomes available that it allows pertinent educational uses. However, it is not because a proposal is based on a technical innovation, and not on an educational requirement, that it is meaningless.

First, some technology-driven applications have encountered success with very few educationally-oriented adaptations. See for example effective uses of communication systems.

Second, technology-driven applications may contribute to the understanding of issues. ITSs are an example of systems based on an emergent technology (knowledge based systems) which created a wave of excitement and a lot of disappointments. The first generation of ITSs was addressed by considering as the central issue the fact that the system should be able to solve the problem submitted to learners and provide a trace of the solution process. Advances related to this problem helped it to be understood that this was not the central issue from an educational point of view. Important distinctions such as the difference between *trace* (report of the problem-solving steps) and *explanation* or the difference between *an expert solution* and *a solution that might make sense for learners* were put to the fore. The importance and difficulties of issues such as *diagnosing learners' actions* or *managing learners' models* were best understood, etc. It may be noticed that this work allowed advances not only in the TEL field, but also in AI and Knowledge Engineering.[4]

Innovation does contribute to the advancement of the field when avoiding the process of producing a technological innovation and an *ad hoc* associated discourse of how interesting this innovation is from an educational point of view, this interest not having been proved and lasting no longer than the time taken for the next innovation to emerge. When the implementation of the approach rather than the approach itself is questioned, conducting long-lasting studies is crucial. This is partly what happened in the case of ITSs, some of these systems, after many years of CS and education co-developments, reaching the stage that allows them to be used in classrooms.

### 2.1.3 Taking into Account Conceptual and Technological Complexity

If technology-driven applications and techno-centrism are indeed important issues, this may correspond to different realities according to (1) the system and setting complexity and (2) the software structuring role. As an example, we contrast hereafter important differences in the issues, dynamics and success criteria for ITSs and ICT-based systems.

Considering ITSs, how "success" is to be investigated is rather precise: the system is a success if it tutors learners in a pertinent way and, consequently,

[4]See for example: Clancey, W.J. (1986). From Guidon to Neomycin and Heracles in twenty short lessons: ORN final report 1979–1985. *The AI magazine*, 7(3), 40–60.

learning outcomes are improved. This requires many different issues to be dealt with: if a solver that can tackle the problems the learners will be presented with cannot be constructed, if the system cannot understand learners' activity and provide scaffolding or if this scaffolding is non-pertinent, the system is unusable. ITSs' teaching features require definitive success on many difficult problems. Such systems cannot be built without considering educational dimensions from the start, and there is little chance that a technology-driven process matches the conditions for success.

Let us now consider systems developed as clever applications of ICT features (synchronous and asynchronous electronic communication; sharing resources; searching for resources; etc.). Such a use of these technologies by actors (computer scientists, educationalists, teachers) having innovative ideas is made easy by the fact that it generally is not necessary to address basic technological dimensions (e.g., network technical issues) but rather how to use services and integrate them in a user-friendly interface. Teachers or learners can make these technologies theirs (e.g., tuning software to their needs), which is unlikely for complex software such as ITSs. Success or failure is therefore much less binary than in the ITS case: the fact that how the different services are integrated in interfaces is not optimal is less an issue than when an ITS unproductively constrains learners or provides non-pertinent scaffolding. Moreover, most of these systems are not meant to teach, but to provide learners with means. As a consequence, "success" may correspond to different dimensions such as the fact that the system generates some interest (from researchers, institutions, teachers or learners), the fact that pioneering ecological experiments may be implemented, or the fact that these experiments turn out to be positive in one way or another (e.g., that they improved learners' motivation). Whether the system "is used" is the killing argument.

This example illustrates that the techno-pushed dimension of the field is to be analyzed taking into consideration the *raison d'être* of systems and their complexity, which lead to different educational/technical interplays. Moreover, the socio-technical dimensions of the field also play an important role (see discussion in Chap. 3, Sect. 3.2).

Many ICT-based systems are as techno-centered as the first uses of CS in education. The difference lies in the fact that the nature of what the technology is about (communication) and the fact that adaptation is easy creates a context that facilitates the process of going from the techno-centered idea to efficient educational software. This process is not always managed, and many systems are just techno-centered innovations. However, this appears less salient than for other types of systems, which is related to the nature of their role and the loose notion of "success" for this type of system.

*De facto*, the technology-pushed approach (as opposed to software elaborated as responses to previously defined problems) often addresses users' interests rather than users' needs. These notions are, however, dynamic: the fact that technology opens some new possibilities creates a context within which new needs may appear.

## 2.2 TEL as a Field Where Some CS Problems Arise

Educational software may be approached by computer scientists as a field that generates CS problems to be solved. Within such an approach, emphasis is on elaborating CS abstractions or implementing models and processes from TEL specifications. This corresponds to addressing CS issues taking their origins in TEL concerns, but disentangled from these.

### 2.2.1 Disentangling CS Dimensions from Their TEL Origins

When considering educational software design and implementation, at a certain level of granularity, problems are disentangled from their original context and turned into purely technical issues. As a matter of fact, programs always end as series of 0s and 1s.

As we have mentioned it Chap. 2, Sect. 3.2 (analysis of SPR/CBPS relationships), the process of going from educational descriptions to a program is not a simple translation from something expressed at a level $n$ to something expressed at a less abstract level $n-1$. There is not a kind of line separating educational concerns and CS concerns. Rather than a line, it is a large zone within which occurs the education/CS interplay, which includes conceptualizing issues, defining pertinent notions, designing artifacts, iteratively testing ideas and artifacts, etc. This zone and the level at which educational software issues become pure CS issues is very dependent on the project and the actor's perspective.

Acting in the education/CS interplay zone may take place at different phases: when imagining how educational ideas or concerns may be implemented; when attempting to turn precise educational ideas or requirements into technical specifications; when analyzing software usage.

A. Newell has powerfully shown, in the case of knowledge based systems,[5] the fact that going from a model defined at the knowledge level to an operational model is generally associated with semantic losses. Analyzing level $n-1$ does generally not allow reconstruction of level $n$: the CS level is related to educational dimensions that have impacted the preceding levels, that of conceptualizing what is to be done and designing the system, but only part of these elements is reified in the system, and only partly reified. This is why maintaining models that allow this connection to be clearly maintained is an absolute requirement for linking usage or learning outcomes to the design (see Chap. 8, Sect. 2).

[5]Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, 18(1), 87–127.

### 2.2.2 Addressing CS Issues

Addressing CS issues as such requires precise specifications, and results are to be analyzed consistently with these specifications.

Let us consider the construction of a piece of software whose objective is, given a chat tool, to categorize learners' interactions as questions, ideas, arguments or counterarguments. This may be decomposed into two sub-problems. First, elaborate the required educational abstractions (e.g., the categories, their characteristics, and the categorization criteria). This problem is both an educational problem (elaborate the meaningful categories) and a TEL problem (elaborating the categorization criteria), taking place in the education/CS interplay zone. Second, elaborate and implement the algorithms applying these criteria and producing the output (the messages classified by categories) from the input (the learners' messages). This second problem is a purely CS matter, which may be addressed independently from its pedagogical rationale, and whose solving leads to some advancement in TEL.

With respect to the piece of software, what is addressed is not "does categorizing sentences helps to scaffold learners?" or a similar educational issue, but "is categorization as here defined implementable" or "what categorization ratio can be obtained?" Such a work may allow results such as "algorithm A allows 82% of messages to be correctly categorized" or "technique B improves categorization by a factor of 0.3". Sticking to the CS dimension does not allow, for instance, comparison of two systems categorizing learners' interactions according to different categories.

Conversely, when the overall project includes a sub-project that consists of a CS issue, such work is not to be evaluated with respect to educational outcomes, but with respect to the technical specifications. The CS work may be perfectly successful from the point of view of the addressed CS objective, independently of whether the underlying learning hypothesis makes sense, the software is used as expected or has the targeted impact. The success of the CS work can be distinguished from the success of the project it is part of.

## *2.3 TEL as a Proper Field*

Finally, educational software may be approached by computer scientists as a proper field, to be addressed as such. Within such an approach, emphasis is on elaborating the abstractions required to understand, design and implement educational software in a way that transcends disciplinary boundaries (transdisciplinarity).

Adopting such a view does not mean that all educational software issues are to be addressed in this way. Rather, it may be developed by acknowledging that some issues are purely educational issues, some others are purely CS issues, but the education/CS interplay zone cannot be addressed by translating or projecting educational issues onto a technical plan: there is not a pedagogical idea on the one side and CS work on the other, but co-constructions.

Within this view, the specific skill of TEL-oriented computer scientists (or, in other words, TEL-specialists with a background in CS) is the capacity to pertinently address the conceptual zone within which occur the education/CS interplays. In the case of innovation, it consists of the capacity of interpreting and analyzing in terms of CBPSs the new possibilities opened by technological advancement. In the case of TEL problems from which CS problems are derived, it consists of the capacity to manage the difficulty of going from a problem expressed at a CBPS level to a problem expressed at a technical level. In this second case, more particularly, this includes participating in the transdisciplinary work of conceptualizing issues and defining pertinent notions, the ability not to fall into naïve interpretations of pedagogical issues and, for instance, a reductive projection of complex issues onto technology. In both cases, it requires the capacity to understand and manage the pedagogical and technical complexity and dynamics, and maintain the correspondence and the traceability between the CBPS-level models and the computable models (see Chap. 8, Sect. 2).

These characteristics typically denote issues that may powerfully be addressed in the light of an engineering view of the field (see Chap. 6).

## 3 Conclusions

In this section, different roles that may be played by CS and the different approaches within which computer scientists may address the TEL field and these roles have been disentangled. These roles and views are not orthogonal to each other and usually co-exist as different dimensions of effective work. They must not be seen as categories to be chosen, but rather as analytical distinctions that may help in making clear and understanding what underlines projects, actions or discourses. Considering such questions is likely to make clearer these features only if, however, it is not conducted in a way that leads to opposing disciplines and addressing in a disciplinary way issues which are intrinsically multidisciplinary.

Concluding this CS analysis of the field, it might be worth pointing out that the disciplinary dimension is both a matter of content and of institutions. Indeed, it is interesting to reflect on the nature of knowledge or theories concerned in the elaboration of a notion or the implementation of a project, what implicit or explicit points of views or representations underlie a given position, etc. However, disciplines are also institutional constructions. Positioning with respect to such dimensions is sometimes related, to a greater or lesser extent, to individual needs to be recognized as a member of a discipline. This of course creates noise and disturbances. As a matter of fact, it is not unusual that researchers focusing on educational software define themselves as such (e.g., “TEL researchers”) and not as “computer scientists concerned in TEL” or “psychologists concerned in TEL”. This is, however, more often the case at a stage of the academic career that does not require, for instance, an institutional position to be obtained.