Evaluating Computer-Supported Learning Initiatives

John B. Nash
Stanford University, USA

Christoph Richter
University of Hannover, Germany

Heidrun Allert
University of Hannover, Germany

INTRODUCTION

The call for the integration of program evaluation into the development of computer-supported learning environments is ever increasing. Pushed not only by demands from policy makers and grant givers for more accountability within lean times, this trend is due also to the fact that outcomes of computer-supported learning environment projects often fall short of the expectations held by the project teams. The discrepancy between the targets set by the project staff and the outcomes achieved suggests there is a need for formative evaluation approaches (versus summative approaches) that facilitate the elicitation of information that can be used to improve a program while it is in its development stage (c.p., Worthen, Sanders & Fitzpatrick, 1997). While the call for formative evaluation as an integral part of projects that aim to develop complex socio-technical systems is widely accepted, we note a lack of theoretical frameworks that reflect the particularities of these kind of systems and the ways they evolve (c.p., Keil-Slawik, 1999). This is of crucial importance, as formative evaluation will only be an accepted and effective part of a project if it provides information useful for the project staff. Below we outline the obstacles evaluation faces with regard to projects that design computer-supported learning environments, and discuss two promising approaches that can be used in complimentary fashion.

BACKGROUND

According to Worthen et al. (1997), evaluation is "the identification, clarification, and application of defensible criteria to determine an evaluation object's value (worth or merit), quality, utility, effectiveness, or significance in relation to those criteria." In this regard evaluation can serve different purposes. Patton (1997) distinguishes between judgment-, knowledge- and improvement-oriented evaluations. We focus on improvement-oriented evaluation approaches. We stress that evaluation can facilitate decision making and reveal information that can be used to improve not only the project itself, but also outcomes within the project's target population. The conceptualization of evaluation as an improvement-oriented and formative activity reveals its proximity to design activities. In fact this kind of evaluative activity is an integral part of any design process, whether it is explicitly mentioned or not. Accordingly it is not the question if one should evaluate, but which evaluation methods generate the most useful information in order to improve the program. This question can only be answered by facing the characteristics and obstacles of designing computer-supported learning environments.

Keil-Slawik (1999) points out that one of the main challenges in evaluating computer-supported learning environments is that some goals and opportunities can spontaneously arise in the course of the development process and are thus not specified in advance. We believe that this is due to the fact that design, in this context, addresses ill-structured and situated problems. The design and implementation of computer-supported learning environments, which can be viewed as a response to a perceived problem, also generates new problems as it is designed. Furthermore every computer-supported learning experience takes place in a unique social context that contributes to the success of an intervention or prevents it. Therefore evaluation requires that designers pay attention to evolutionary and cyclic processes and situational factors. As Weiss notes, "Much evaluation is done by investigating outcomes without much attention to the paths by which they were produced" (1998, p. 55).

For developers designing projects at the intersection of information and communication technology (ICT) and the learning sciences, evaluation is difficult. Evaluation efforts are often subverted by a myriad of confounding variables, leading to a "garbage in, garbage out" effect; the evaluation cannot be better than the parameters that were built in the project from the start (Nash, Plugge & Eurlings, 2001). Leaving key parameters of evaluative thinking out

of computer-supported learning projects is exacerbated by the fact that many investigators lack the tools and expertise necessary to cope with the complexity they face in addressing the field of learning.

We strongly advocate leveraging the innate ability of members of the computer science and engineering communities to engage in "design thinking" and turn this ability into a set of practices that naturally becomes program evaluation, thereby making an assessment of the usefulness of ICT tools for learning a natural occurrence (and a manifest activity) in any computer-supported learning project.

Design-Oriented Evaluation for Computer-Supported Learning Environments

There are two approaches that inherently relate themselves to design as well as to evaluation. Therefore they are useful tools for designers of computer-supported learning initiatives. These two perspectives, discussed below, are scenario-based design and program theory evaluation. Both approaches assume that the ultimate goal of a project should be at the center of the design and evaluation discussion, ensuring a project is not about only developing a usable tool or system, but is about developing a useful tool or system that improves outcomes for the user. Beyond this common ground, these approaches are rather complementary to each other and it is reasonable to use them in conjunction with one another.

Scenario-Based Approaches

Scenario-based approaches are widely used in the fields of software engineering, requirements engineering, human computer interaction, and information systems (Rolland et al., 1996). Scenarios are a method to model the universe of discourse of an application, that is, the environment in which a system, technical or non-technical, will be deployed. A scenario is a concrete story about use of an innovative tool and/or social interactions (Carroll, 2000). Scenarios include protagonists with individual goals or objectives and reflect exemplary sequences of actions and events. They refer to observable behavior as well as mental processes, and also cover situational details assumed to affect the course of actions (Rosson & Carroll, 2002). Additionally it might explicitly refer to the underlying culture, norms, and values (see Bødker & Christiansen, 1997). That said, scenarios usually focus on specific situations, only enlighten some important aspects, and generally do not include every eventuality (e.g., Benner, Feather, Johnson & Zorman, 1993).

Beside their use in the design process, scenarios can also be used for purposes of formative evaluation. First of all, as a means of communication, they are a valuable resource for identifying underlying assumptions regarding the program under development. Stakeholder assumptions might include those related to instructional theories, the learner, the environmental context, and its impact on learning or technical requirements. Underlying assumptions such as these are typically hidden from view of others, but easily developed and strongly held within individuals developing computer-supported learning environments. Scenarios help to reveal the thinking of designers so that others can participate in the design process and questionable assumptions can come under scrutiny. The use of scenarios also allows identification of pros and cons of a certain decision within the design process. In this vein Carroll (2000) suggests employing "claim analysis." Claims are the positive or negative, desirable and undesirable consequences related to a certain characteristic of a scenario. Assuming that every feature of a proposed solution usually will entail both positive and negative effects helps to reflect on the current solution and might provoke alternative proposals. The analysis of claims is thereby not limited to an intuitive ad hoc evaluation, but also can bring forth an explicit hypothesis to be addressed in a subsequent survey.

Program Theory Evaluation

Program theory evaluation, also known as theory-based evaluation, assumes that underlying any initiative or project is an explicit or latent "theory" (or "theories") about how the initiative or project is meant to change outcomes. An evaluator should surface those theories and lay them out in as fine detail as possible, identifying all the assumptions and sub-assumptions built into the program (Weiss, 1995). This approach has been promoted as useful in evaluating computer-supported learning projects (Strömdahl & Langerth-Zetterman, 2000; Nash, Plugge & Eurlings, 2001) where investigators across disciplines find it appealing. For instance, for designers (in mechanical engineering or computer science), program theory evaluation reminds them of their own use of the "design rationale." And among economists, program theory evaluation reminds them of total quality management (TQM). In the program theory approach (Weiss, 1995, 1998; Chen, 1989; Chen & Rossi, 1987), one constructs a project's "theory of change" or "program logic" by asking the various stakeholders, "What is the project designed to accomplish, and how are its components intended to get it there?" The process helps the project stakeholders and the evaluation team to identify and come to consensus on the project's theory of change. By identifying and describing the activities, outcomes, and goals of the program, along with their interrelationships, the stakeholders are then in position to identify quantifiable measures to portray the veracity of the model.

Theory-based evaluation identifies and tests the relationships among a project's inputs or activities and its outcomes via intermediate outcomes. The key advantages