Improving Evaluations in Computer-Supported Learning Projects

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 groups and grant makers who desire greater accountability in lean times, this trend is due also because 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 (vs. summative approaches) that derive information that can be used to improve a program while it is in its development stage (see Worthen, Sanders, & Fitzpatrick, 1997). And in spite of the known benefits of integrating evaluation into the project development process, we note a lack of theoretical frameworks that reflect the peculiarities of computer-supported learning projects and the ways they evolve (see 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. The purpose of this chapter is to outline the obstacles to integrating evaluation in computer-supported learning projects and then discuss two promising approaches that can be used to address these challenges.

BACKGROUND

According to Worthen, Sanders and Fitzpatrick (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 explicitly mentioned or not. Accordingly, it is not the question if one should evaluate, but which evaluation methods generate the most useful information 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, thus, are not specified in advance. We believe that this is because 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 often are 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

Two approaches 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, scenario-based design and program theory evaluation, are discussed below. 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 only about developing a usable tool or system, but also 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.

SCENARIO-BASED APPROACHES

Scenario-based approaches are widely used in the fields of software engineering, requirements engineering, human computer interaction, and information systems (Rolland, Achour, Cauvet, Ralyté, Sutcliffe, Maiden, Jarke, Haumer, Pohl, Dubois, & Heymans, 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 the underlying culture, norms and values (see Bødker & Christiansen, 1997). That said, scenarios usually focus on specific situations and only enlighten some important aspects; they generally do not include every eventuality (e.g., Benner, Feather, Johnson, & Zorman, 1993).

Besides 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