

Chapter Proposal for Online Collaborative Learning: Theory and Practice

**Drawing on Design to Improve Evaluation of  
Computer Supported Collaborative Learning:  
Two Complementary Views**

November 15, 2002

John B. Nash, Stanford University  
Stanford Center for Innovations in Learning  
450 Serra Mall, Building 160  
Stanford, CA 94305 USA  
Voice +1 650 924 0152  
Fax +1 650 725 5916  
[jnash@stanford.edu](mailto:jnash@stanford.edu)

Christoph Richter, University of Hanover  
Learning Lab Lower Saxony [L3S]  
Deutscher Pavillon 1.OG  
Expo Plaza 1  
D- 30539 Hannover  
Voice +49 (0)511 762 9754  
Fax +49 (0)511 762 9779  
[richter@learninglab.de](mailto:richter@learninglab.de)

Heidrun Allert, University of Hanover  
Learning Lab Lower Saxony [L3S]  
Deutscher Pavillon 1.OG  
Expo Plaza 1  
D- 30539 Hannover  
Voice +49 (0)511 762 9756  
Fax +49 (0)511 762 9779  
[allert@learninglab.de](mailto:allert@learninglab.de)

NOTE: The authors wish to acknowledge Mark Painter, Technische Universität Braunschweig, for agreeing to offer his project, MoCA, as a case to examine for this chapter.

## **A. Objectives or Purposes**

This chapter will address theoretical frameworks of the evaluation of learning technologies. We examine two approaches for design and evaluation of computer support for collaborative learning (CSCL) and then proffer ways in which the two approaches can be used together to form a compelling approach to improving design and evaluation of CSCL experiences. Our perspectives on these matters are based in part on experiences as directors of evaluation in laboratories in the United States and Germany within the Wallenberg Global Learning Network (WGLN). Our collective experience includes work on technology and teaching reform in higher education, program evaluation theory, evaluation of CSCL, learning design, and development of learning objects, open learning repositories and the semantic web.

The WGLN was launched in 1998 with the support of the Knut and Alice Wallenberg Foundation and the Marcus and Marianne Wallenberg Foundation in Stockholm, Sweden. The Stanford Learning Laboratory at Stanford University, and the Swedish Learning Lab (with member institutions the Royal Institute of Technology (KTH), Uppsala University, and the Karolinska Institute) were charter members of the WGLN. In 2000 the Learning Lab Lower Saxony (L3S) joined the WGLN, bringing with it partner universities the University of Hanover, Braunschweig Technical University, Braunschweig School of Art, University of Karlsruhe, University of Mannheim as well as a host of research institutes. Committed to improving formal and informal learning and tackling the challenges facing universities in teaching with technology, the partners of the WGLN oversee a targeted grants program that is committed to funding collaborative international research and development in technology-supported pedagogical practices.

In 2001-2002 the WGLN funded investigators who examined the following:

- Web-based tools for project and team development;
- Tool for high level individual feedback for language learners;
- Interactive study material and computer-based tools for conceptual modeling;
- Software tools for knowledge management;
- Web-based prescription tool for medical students and health professionals;
- Prototypes for 3D interaction and visualization;
- Introduction of pedagogically innovative computer applications;
- Simulation of virtual patients for problem-based learning for medical students;
- Life Sciences multidisciplinary dynamic courselets/modules.

The WGLN expects investigators to derive an evaluation plan that links their work to WGLN (2002) goals, which are to:

- Develop learners' cognitive, social, and professional proficiency through engaging and motivational learning experiences;
- Improve learning at school, at work, in the family, and within society;
- Produce new knowledge and define requirements for best learning practices and individualized learning approaches; and
- Provide pedagogic and technical solutions suitable for innovative use in a variety of settings.

A challenge facing the WGLN, and the CSCL field in general, is the fact that investigators and developers are often experienced scientists, yet novice evaluators. The CSCL field increasingly attracts a large number of researchers from the humanities and sciences into projects that are designed to improve learning—matters these researchers heretofore have not been asked to address. Data from an implementation evaluation conducted in the first half of

academic year 2001-2002 on the thirty projects funded by the WGLN (WGLN, 2002) suggested there was a great deal of variance in the investigators' ability to situate their work within educational theory. Some investigators were able to be specific in describing the educational theories underpinning their work, while others merely used very high-level buzzwords such as "problem based learning" or "constructivist learning" without any real connection back to the literature. Most proposal authors did not place their work within a higher-level theoretical context related to pedagogy or cognition nor gave compelling evidence that they understood educational theory when they alluded to it.

We believe it is important to encourage CSCL developers to consider the overall context in which software will be developed and operated, that is, the *universe of discourse* (Leite et al, 2000). There is growing recognition that the development of new e-learning systems is intertwined with the development or reorganization of the learning situation where the new system is intended to be used (Janneck, 2002). Baumgartner (1997) states that pedagogical and educational evaluation must not be restricted to the software itself but also has to focus on the social situation where the software will be used. The idea is that the primacy of the learning environment should reign over the technical artifact. Reductionism that creeps into projects and thereby lends emphasized focus on the technical system is not appropriate because as Carroll (2000, p. 46) puts it: "They (the computers) unavoidably restructure human activities, creating new possibilities as well as new difficulties." By considering the universe of discourse, developers must think clearly about the educational outcomes enhanced by a computer-supported solution and not just the usability of the computer support.

For investigators executing projects in information and communication technology (ICT) and the learning sciences, program evaluation is difficult. Program 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 et al. 2000). Our experience in the WGLN suggests that most investigators lack the tools and expertise necessary to specify how to measure whether goals have been met.

A typical proposal is largely devoted to planning the program or the technical support of computer supported collaborative learning situations. Furthermore, the purpose of the project is often restricted to the design phase. Therefore it is of great importance to stress the contributions of evaluation approaches to the planning of programs. It becomes crucial to anticipate possible outcomes and side effects as early as possible.

Further findings from the implementation evaluation conducted in 2001-2002 (WGLN, 2002) suggested that project teams with computer science and engineering backgrounds underestimated the time needed to conceive and conduct fruitful evaluation work. Project staffs often plan evaluations independently from other project activities and restrict the evaluation work to a limited frame of time, usually far off in the future. This is not to be unexpected considering the notion that most computer scientists or electrical engineers are not trained in these topics. Lack of experience, and therefore lack of appreciation as to what evaluation entails, contributes, we allege, to the detrimental postponement of evaluation activities

In this paper we leverage the language 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 pedagogical utility of these tools into a natural occurrence (and a manifest activity) in any CSCL project.

### **Perspectives or Theoretical Frameworks**

The perspectives discussed here are scenario-based design and program theory evaluation.

### Scenario-Based Design.

Scenario-based approaches are widely used in the fields of software engineering, requirements engineering, human computer interaction, and information systems (Rolland et al. 2001). Scenarios have also been used for purposes of strategic planning in fields as business, economics, politics (e.g., Kahane, 2000).

Scenarios are an approach to model the universe of discourse of the application, that is, the environment in which the system will be deployed (Breitman and Leite, 2001). A scenario can be an important resource for eliciting requirements for an application's design and evaluation. But the use of scenarios is not limited to describing the artifact and its use. They are also useful to describe any system or interpersonal process that is situated in a complex environment (see Benner et al., 1993).

For the purposes of our research we refer to a scenario as a concrete story about use of an innovative tool, software, etc. (Carroll 2000). Scenarios include protagonists with individual goals or objectives and reflect exemplary sequences of actions and events.

Rosson and Carroll (2002, p. 18) list the following characteristic elements of user interaction scenarios:

- Setting. Situational details, that motivate or explain goals, actions, and reactions of the actor(s)
- Actors. Human(s) interacting with the computer or other setting elements; personal characteristics relevant to scenario
- Task goals. Effects on the situation that motivate actions carried out by actor(s)
- Plans. Mental activity directed at converting a goal into a behavior
- Evaluation. Mental activity directed at interpreting features of the situation

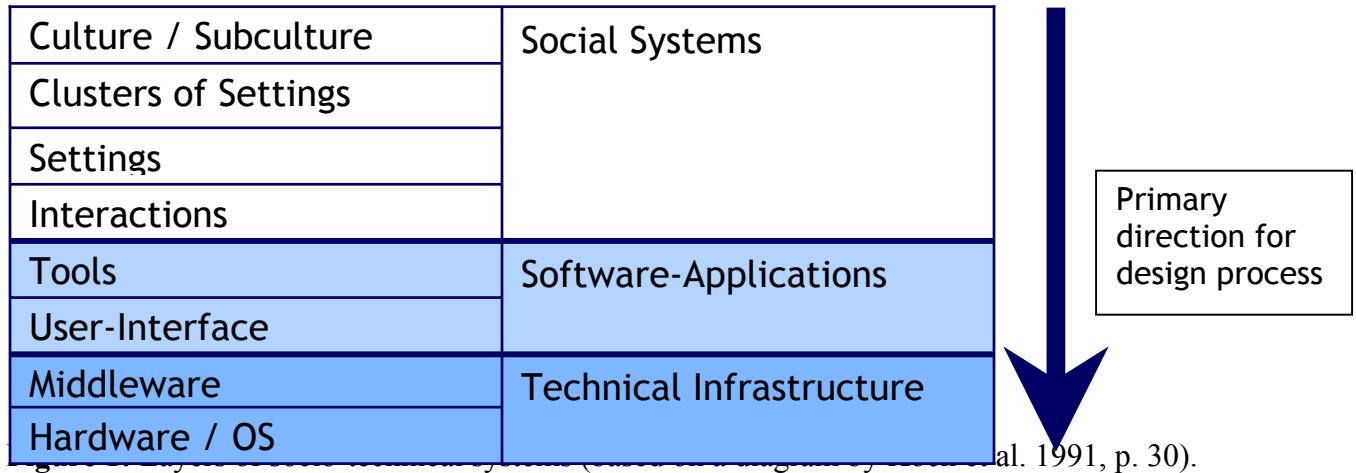
- Behavior. Observable behavior
- Events. External actions or reactions produced by the computer or other features of the setting; some of these may be hidden to the actor(s) but important to scenario.

The following scenario shows how some of these characteristic elements might interact in an imagined situation. While a scenario can contain all of the characteristics listed above, not all of them are expressed explicitly in the following example.

*Tom is a student of computer sciences at the University of Hanover. He is attending a introductory lecture about artificial intelligence. The lecturer of this course asks the students to use a workspace in order to share their homework. Tom has a first look at it that night. He explores some of the functionalities: one can upload documents and anyone participating in the course can annotate these documents and write comments. "Ugh," he considers, that means fellow students as well as the lecturer can write comments which are visible to all? Does that mean he has to share his mistakes and imperfect work? He is displeased with the system. Up to now he felt the atmosphere in this course was quite competitive and not at all collaborative!*

As the story about Tom demonstrates, scenarios focus on specific situations and only enlighten a few important aspects rather than drawing a complete picture of the universe of discourse (e.g., Benner et al., 1993). At the same time scenarios include characteristics that refer to different layers of socio-technical systems. This feature is of great importance while a project is involved in designing a computer supported learning environment. Besides technical aspects, the learning environment is inevitably embedded in a social system which is shaped by instructional ideas as well as organizational requirements. For our purpose scenarios are not

restricted to human computer interaction but might include the whole range of social processes the learner is involved in. Figure 1 shows a diagram reflecting different layers that have an impact on socio-technical systems.



Several ways have been described to reveal scenarios. Scenarios can be elicited via ethnographic field study, participatory design, reuse of prior analyses, scenario typologies, theory-based scenarios, technology-based scenarios, and brain storming techniques (Carroll, 2000).

To date there has been no systematic adaptation of the scenario-based approach to the field of design and evaluation of computer-supported learning environments. Here we show the role of scenarios within the design and evaluation of computer-supported learning environments. There are different aspects to how scenarios support a project.

Scenarios provide a common level of communication among the different stakeholders of the project.

Scenarios, in the form of narrative stories, are easy to produce and understand. Describing the universe of discourse at the level of concrete user activity allows every stakeholder to communicate her or his ideas about an educational environment, instructional ideas, and

technical constraints without being an educationalist or a software engineer, relying on one's own theories and jargon. However, it is easy for any specialist to interpret a scenario in terms of her or his domain and identify familiar terms and theories.

It is crucial for any project team to model the universe of discourse, to get a common understanding of the "thing" to be designed. Scenarios are useful for designing and evaluating innovative learning environments that are supported by technology. As explained in the introduction, the description of context is crucial within the design of any socio-technical system such as learning environments.

According to Koch et al (1991), the design of computer-supported systems, like the scenarios describing them, has to start on social and organizational levels. Later within the progress of the project, scenarios also address the technology that supports learning and teaching. Scenarios help to make assumptions explicit. For fruitful use of technology, context and technological support should go well together. If inconsistency occurs, it can be tackled by either changing context at the social and organizational level or to change technology. In the scenario presented above, the competitive atmosphere may be changed and transformed to a more collaborative one. But it is also imaginable to change the technology by reducing collaborative features or at least to make comments in a private mode.

Scenarios, in the form of narrative stories, can be used to describe learning situations on every level of granularity.

Granularity can range from describing organizational setting of a course to specific computer-supported tasks a student might accomplish within a course. Scenarios provide examples of what *could* occur in a learning environment, rather than complete pictures of *all* aspects of the learning environment. Because of being selective, it is important to identify a set

of diverse scenarios that highlight specific aspects of the learning situation. Existing scenarios can also be a starting point to find new scenarios. To obtain a more systematic impression of the universe of discourse, Bødker and Christiansen (1997) suggest the use of checklists with questions as a way to help clarify and complete the scenarios. For example: in the work-oriented checklist, which includes 14 items, one item deals with materials and outcomes. Each item comes along with a short definition and with some questions: “What kind of materials (documents, pictures, drawings etc.) are involved in the production of which kind of products?” (Bødker, Christiansen und Thüring, 1995). The checklists enable the stakeholders “to find out relevant constraints and key-concerns” (Bødker et al, 1995).

While Bødker et al (1995) presented a work-oriented and a technical checklist, a checklist applicable for learning situations is not yet available but might be of great use. Checklists that cover technical, educational and organizational aspects of proposed computer-supported learning environments will help to link the planned learning environment to its contextual setting. Inevitably the scenarios will have to be described on different levels of granularity. Technical aspects at one end of the granularity scale relate to questions of the overall technical infrastructure. At the other end of the granularity scale they relate to Input-Output devices. Organizational aspects at one end of the granularity scale relate to possible changes in a whole institution that might be activated by the introduction of new learning scenarios. At the other end of the granularity they relate to efforts to be accomplished by a single person to manage different tasks he is involved in.

Scenarios help to identify and make explicit underlying assumptions of the stakeholders.

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 held strong within individuals developing learning environments. Tacitly held, these assumptions can remain untested or worse yet, plausible but unfounded. Underlying assumptions that underpin rationales in computer-supported learning environments can be drastically oversimplified by designers if the assumptions are not exposed in a scenario. It is easy to convince one's self (that is, accept an underlying assumption) that "if knowledge is organized systematically learning will be much easier." But without having considerable knowledge of the cognitive science and educational research literature, the capabilities of a computer supported learning environment to systematically organize knowledge could end up reflecting the presuppositions of a computer scientist or engineer striving for usability. Scenarios help reveal the thinking of designers so that others can participate in the design process.

Other common examples of assumptions we've seen in our work include:

- "the learner has to be reinforced externally in order to motivate him or her";
- "the context is of importance";
- "every learner has access to the internet free of charge";
- "the use of metadata facilitates searching and finding Learning Objects".

If these assumptions, hypothesis and expectations are not expressed explicitly they might influence the design process in an unpredictable way or might provoke misunderstandings between the stakeholders. Technology development builds on explicit and implicit assumptions..

If an investigator recognizes the importance of context, this might lead to a discussion regarding the effects of a certain technological feature or aspect of the social environment has the

intended effect. Or it might even lead to the constitution of a research question in order to test whether context actually is of importance.

Scenarios can also be used to identify the pros and cons of a certain decision within the design process.

Many decisions are made during the design process of a computer-supported learning environment, some of which can be tested and revised. Other decisions once reached cannot be undone within a project because of limited resources or other constraints. In these cases it is critical to discuss the possible implications in advance, especially as they relate to usability. These are questions related to prospective or anticipatory evaluation (c.f. Wottawa & Thierau, 1998).

The assumption “The use of metadata facilitates search and find of Learning Objects” may lead to the decision to annotate Learning Objects with metadata. But it is also possible that metadata distracts the students from other relevant material (which is not annotated). Only explicit decisions can be tested whether they lead to the intended effect. To know the underlying assumptions of a certain decision helps to develop hypothesis why the decision lead to the intended effect or why not.

For this purpose Carroll (2000) suggests employing “claim analysis.” Claims are the positive or negative, desirable and undesirable consequences related to a certain characteristic of a scenario. Other methods from the field of strategic planning might also be employed in order to enlighten possible consequences of a certain decision.

Beside all this scenarios help the stakeholders not to forget the persons for whom they design the educational setting. Scenarios do so by describing the universe of discourse at the level of concrete user activity, as discussed above. . Human actors are an important factor to take into account when designing an educational setting. Without considering them, the design of software

that is meant to support learning processes is often seen as developing “a system”. However, in practice questions like: “how should we design the system architecture”, can easily dominate the discourse. According to Nygaard (1986), a designer who adopts this system perspective will interpret the universe of discourse as consisting of “data flows”, “transactions”, “records”, “relations”, “objects”, etc.. The Unified Modeling Language (c.f. Rumbaugh et al., 1999) that is prominent in the field of computer science also reflects this system perspective. While “the system is the ‘complete model’” a use-case describes “a sequence of actions [...]that a system [...] can perform by interacting with out-side actors” (Rumbaugh et al., 1999). Use-cases usually just focus on the observable behavior occurring in the user system interaction while cognitive processes, prior experience, organizational constraints stay implicit. (cp. Carrol, 2000). While the system perspective in general and use-cases in particular are very important for the design of useful software artifacts they are insufficient to describe educational settings and learning processes.

The description of the intended or actual learning situation is an essential prerequisite for the requirements elicitation of the technical system. Furthermore the specific requirements for a technical system vary with the concrete learning situation where it is used. Because of this it is crucial for the design of the technical system to reveal the situational factors that might have impact on the system and its use.

We now turn our discussion to the second perspective we wish to address, program theory evaluation.

### **Program Theory Evaluation**

Program theory evaluation assumes that underlying any intervention is an explicit or latent “theory” (or “theories”) about how the intervention is meant to change outcomes (Granger,

1999). 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 CSCL 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.” Among the economists we have talked to, program theory evaluation reminds them of total quality management (TQM). In the program theory approach (Weiss, 1972, 1995, Chen & Rossi, 1989), one constructs the project’s “theory of change” or “program logic” by discussing with various stakeholders, "What is the project designed to accomplish, and how are its components intended to get it there?" Also known as a theory-based evaluation strategy, 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.

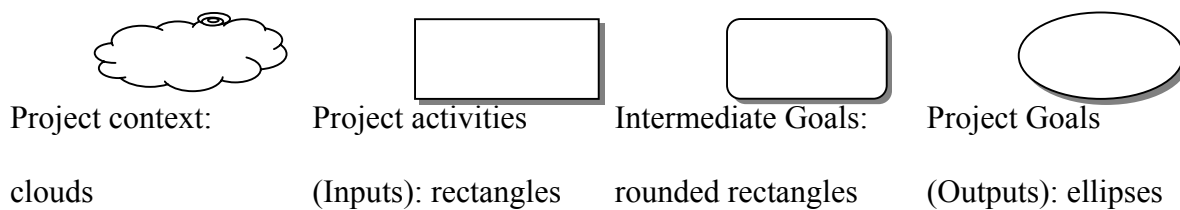
Theory-based evaluation identifies and tests the relationships among a project’s inputs or activities and its outcomes via intermediate variables. The key advantages to using theory-based evaluation include (Connel & Kubisch, 1995; Weiss, 1995):

- It asks project practitioners to make their assumptions explicit and to reach consensus with their colleagues about what they are trying to do and why.
- It articulates a theory of change at the outset and gains agreement on it by all stakeholders reducing problems associated with causal attribution of impact.
- It concentrates evaluation attention and resources on key aspects of the project.

- It facilitates aggregation of evaluation results into a broader context based of theoretical program knowledge.
- The theory of change model identified will facilitate the research design, measurement, data collection, and analysis elements of the evaluation.

In our research, program logic maps are used to develop and communicate program theory. The program logic map provides a graphical description of the initiative or project, the intended outputs, and the intended outcomes as defined by the intervention's theory of change. Program logic maps show temporal sequences, building left to right, and portraying relationships with arrows. Program logic mapping forces project teams to make an explicit statement of the relationship between project activities (inputs) and project goals (outputs). The map (and the mapping process itself) is then used to plan an evaluation, develop indicators, choose instrumentation, and decide when indicators should be measured.

A program logic map is comprised of four basic shapes: clouds describing the project context, rectangles describing inputs, rounded rectangles showing intermediate goals, and ellipses representing ultimate goals (figure 1).

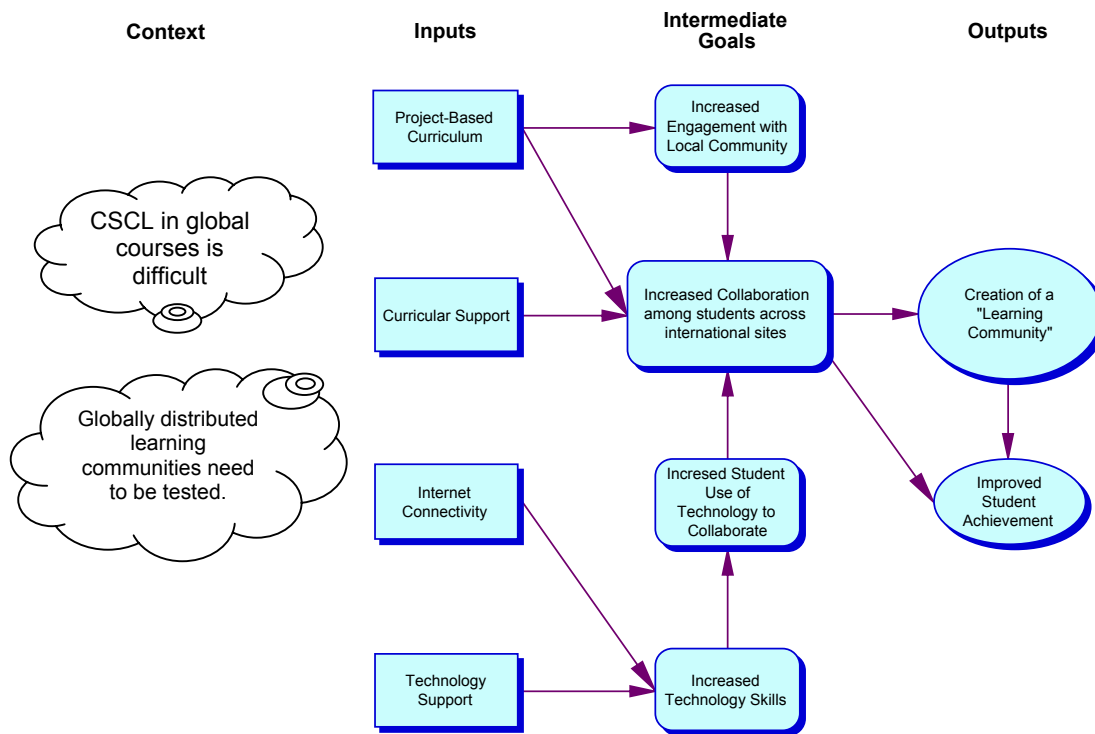


**Figure 1.** Program logic map shapes.

From a program logic map one can derive a succinct summary statement of the project’s logical path. What follows is an example based upon the sample program logic map shown in figure 2.

*Given that it is difficult to engender a community of learners in geographically distributed courses using CSCL, a course offered to college students in geographically*

*disparate sites, is theorized to operate as follows: a project-based community curriculum, coupled with curricular support will lead to increased engagement in the local community by students and increased collaboration among students across communities. Internet connectivity and technology support will lead to increased technology skills on the part of the student thus increasing their ability to collaborate. Increased collaboration is theorized to lead to the creation of a learning community and therefore improved student outcomes.*



**Figure 2.** Sample program logic map.

Developing the statement is merely a linking of the elements of one's map in a way that makes sense to the reader. Developing a statement of the project's logical path provides a checking mechanism on the design of the initiative.

## **B. Modes of Inquiry and Source of Evidence**

To illustrate these approaches we highlight a project funded in 2001-2002 by the WGLN was selected as the source for data from which scenarios and program theory could be developed. The authors met with the project's investigator to develop project scenarios (Richter and Allert) and the project's program theory (Nash). The scenarios and program theory were developed in separate sessions with the project investigator. The mode of inquiry went as follows:

- Discussion with the investigator about his project, its design, and ultimate goals.
- Development of either a scenario or program theory (depending on the session) with the investigator
- Discussion with the investigator about
  - Differences in how the project is conceived of after having developed scenarios and a program theory
  - Questions that have arisen regarding project direction, goals and quality that were not evident prior to the process
  - Matters related to the quality of operational definitions of key variables in the project before and after the development of scenarios and program theory

### **Data Source: The Modular Content Archives Project**

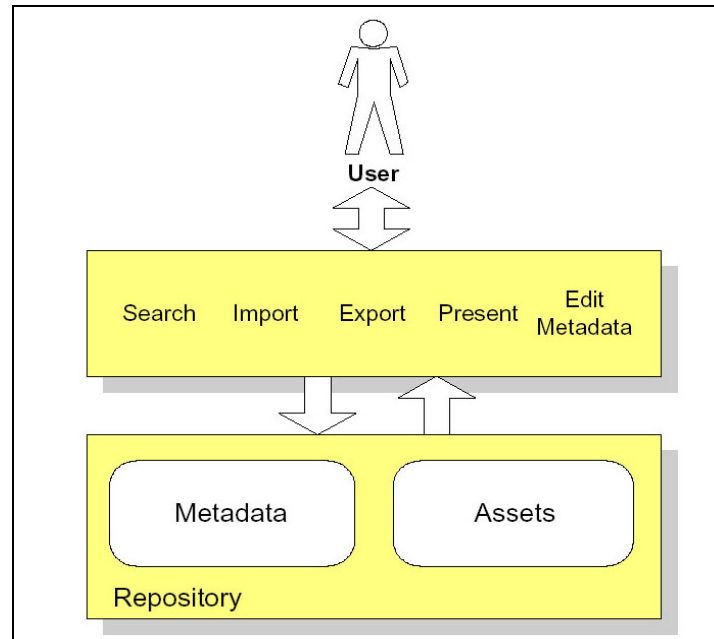
Modular content archives (MoCA) project is a sub-project within the Personalized Access to Distributed Learning Repositories (PADLR) project at the University of Hanover in Germany and receives fiscal support from the WGLN. MoCA is classified within a subset of PADLR referred to as 'server and client side tools', which focuses on modular learning environments and modular material collections, as well as tools for video/audio capturing and annotation using metadata.

Within the MoCA project various software tools are under development, that aim to support intelligent and flexible archiving, management, allocation, and distribution of modular content. Modular Content refers to lecture slides, presentations, scientific graphics and texts, description of experiments, software, animations, simulations, and so on. The intended software tools will help to simplify the re-use of content for different purposes and different audiences. The designers need to be aware of potential problems to enable intelligent archiving and flexible and intuitive access to modularised content. For example:

- Content modules are often detached from context and therefore can hardly be found,
- Content modules can be of different file types,
- Need for version lists.

(for more details see PADLR (2001)).

One of the software tools under development in the MoCA project is the Repository for flexible access to modular content (Reflect). Reflect is proposed to be a local database system that supports the archiving, management, allocation, and distribution of modular content by teachers and students. As a peer of the Edutella network (Nejdl et al. 2001) it also will facilitate the exchange of modular content. The basic structure of Reflect is shown in figure 3.



**Figure 3:** Block Diagram of the described application (Painter, 2002)

Reflact “consists of two basic blocks, the repository containing all assets and the corresponding meta-data and the front-end which provides the desired functionality. The user can search the repository for specific assets, can import and export assets or edit meta-data that will be presented in an intuitive way” (Painter, 2002).

Furthermore Painter (2002) names the following requirements for this software tool:

- independency of the data format
- ability to share the documents over a network (internet)
- supporting existing meta-data standards
- possibility to enter and append meta-data
- facilitation of flexible text or metadata search
- lightweight solution
- supporting representation of relationships between content modules (version control, multilanguage, context, prerequisites)

- intuitive user interface.

Faculty and staff plan to use, test and evaluate Reflect within a lecture (a semester-long course) about Digital Communications held at the Braunschweig Technical University.

### **Using Scenarios in the Modular Content Archives Project**

The two scenarios presented here were written in a very early phase of the project and are a sample of six scenarios. Of the six scenarios developed, three scenarios dealt with problems and deficiencies in the present university processes occurring without the Content Browser and the other three show how the Content Browser can help to solve these problems (see Painter, 2002)

#### **First Scenario: Responding to students' questions without the Content Browser<sup>1</sup>**

*Prof. Patao is an instructor of the lecture Digital Communications. He teaches students key issues of the design of digital communication systems, which involves a lot of math. Prof. Patao knows that some concepts are not easy to understand and to facilitate this one of his assistants has developed some simulations that are installed on a PC dedicated to lectures, seminars and so on. The students attending his lecture have a heterogeneous backgrounds since they are coming from five different programs of study. Their prerequisites concerning signal and system theory, which is essential for understanding, is very uneven. One day during a lecture, a student is asking a question about material that is rather a prerequisite for this course. And then Prof. Patao imagines that, in order to answer that question with appropriate and instructive material, he should have to access his PCs hard disk for a simulation that will give an answer very quickly. But he does not have access to his PC in that right moment so he has to use the*

*chalk board to draw an outline. With the simulation that is available somewhere the student would have gained more understanding.*

### **Second Scenario: Preparing for the exam with the Content Browser**

*While preparing for the exam in Digital Communications Eric remembers he has integrated the model to simulate the 64-QAM several weeks ago in his personal portfolio. He starts his computer and accesses his personal portfolio on the server. There he has organized all different kinds of material he obtained during this lecture. He immediately finds the simulation since he organized everything properly using the supported metadata. The search functionality allows him to filter only specific metadata fields, for example the `Format` field that is included in the `Technical` category of the LOM metadata standard. After the tool presents him a list of choices he downloads a copy of simulation model and opens it with the related simulation tool.*

The scenarios were part of the first technical report (Painter, 2002). The purpose of writing scenarios in this stage of the project was to express the motivation for the project. This is one possible range of use for scenarios. We demonstrate further purposes of scenarios: they can also be used by the projects for guiding design and evaluation. In the following we will focus on some of the aspects within the scenarios presented above.

#### Detecting additional scenarios

The first scenario refers to ‘*the lecture Digital Communications*’. It raises the question: “What is unique about this lecture and what are the common aspects?” The scenario includes additional hints that give a first impression of the characteristics of this lecture, for example: ‘*Prof. Patao knows that some concepts are not easy to understand and to facilitate this one of his*

---

<sup>1</sup> The titles were not part of the original scenarios. We also made minor changes to the form of the scenarios.

*assistants has developed some simulations*'. Can project stakeholders assume that for every class topic simulations or other electronic resources are available? Do similar problems also arise in seminars, practicals etc.?

The second scenario can also be used to detect additional scenarios. *'Preparing for the exam'* sets the scenario in a certain context. Scenarios that focus on the characteristics of preparing for exams might lead to a broader understanding of the situation addressed in the given scenario. On the other hand the project team can also think about other student tasks, such as collaborative problem solving, and reflect the special needs of such tasks and how and if they could be supported by the content browser. The detection of additional scenarios helps to prevent the project from focusing on special situations and neglect others also of potential importance.

#### Identifying underlying assumptions, hypothesis etc.

Assumptions on learning and teaching guide the design of both, technology as well as the context as a whole. One of the assumptions that can be identified in the second scenario is: *'It is useful for students to have access to additional electronic resources while preparing for exams, and they will actually use them appropriately.'* On the one hand it can be argued that the additional resources help to achieve a better understanding of the topic. On the other hand using information spaces learners must integrate units of information into a coherent mental representation. This coherence formation process makes great demand on learners' cognitive and metacognitive skills. They must orient themselves and build up connections among single concepts, learning objects, units and courses. They have to relate important items of content. In navigation in non-linear data bases learners suffer from conflicting and competing goal intentions as well as from cognitive overload if the navigational task consumes too much of their resources. Students therefore may want to focus on some recommended basic literature.

An assumption that might underlie the first scenario is: *'With the simulation that is available somewhere the student would have gained more understanding'* in contrast to the outline drawn on the chalkboard. Discussing the pros and cons of both approaches might lead to a deeper understanding of the situation. The identification of underlying assumptions, hypothesis helps to avoid biases. At the same time only identified assumptions and hypothesis can be tested.

#### Identifying pros and cons of the proposed solution

Any proposed as well as implemented solution does not only entail advantages but also disadvantages. Referring to the given scenario: On the one hand the use of electronic resources in the classroom can be helpful because information stored in the database is carefully elaborated. On the other hand prefabricated information units might only roughly fit in the concrete learning situation and might fail to answer the given question in particular. Chalkboard drawings are often not so well elaborated, but they can be adapted to the concrete situation more easily. On the basis of identifying pros and cons the project team can decide if they only want to promote one alternative or if they will look for a solution that combines the advantages of both approaches.

#### Requirements elicitation

The second scenario addresses another important aspect: *'The search functionality allows him to filter only specific metadata fields, for example the Format field that is included in the Technical category of the LOM metadata standard.'* While both topics *'managing electronic resources'* and *'metadata'* are explicitly mentioned in the technical report (Painter, 2002) the scenario focuses on the interrelation of both topics. As a consequence it appears to be relevant to work out a solution of how the students will get familiar with educational metadata and can make adequate queries. While requirements derived from different sources are often presented in an

unconnected manner, scenarios facilitate the reflection on the interrelation of different requirements.

### Evaluation

The questions raised above might be seen as an integral part of project evaluation by themselves. Within this very early stage of the project relevant aspects for evaluation are yet identified. The stakeholders of the project might be interested in evaluating whether the Content Browser is used in the way it is hypothesized in the scenarios. A further elaboration of the scenarios might also help to identify what variables could mediate the use of the Content Browser. Scenarios can be a starting point for formative as well as summative evaluations.

### Reflecting on the problems addressed by the project

While an additional scenario points out how the problem described in the first scenario can be solved by the use of the Content Browser the scenario can also be a starting point to reflect on the projects genuine objectives. This can be done by asking if there are alternative ways to cope with the identified problem: Are there alternative solutions of dealing with students heterogeneous backgrounds and questions that are beyond the scope of the actual course. This might help to prevent the so called 'garbage in, garbage out' situations.

### General issues the use of scenarios reveals

The scenarios were written in a very early project stage. Therefore the set of scenarios as shown above is not complete and additional scenarios might lead to further insights. Also a more detailed analysis of the scenarios would be helpful. While we recognize these shortcomings several important things became obvious during this process.

First, even a small set of scenarios forces project staff into a mode of thinking where developing additional scenarios is needed. Second, scenarios reveal, in often times stark detail,

underlying assumptions and hidden hypotheses held by the investigator or project staff. Third, by placing hypothetical users in a scenario with the computer support tools to be developed, pros and cons of the proposed solution can be discussed and possible effects can be evaluated. Fourth, previously unknown feature requirements can be revealed when scenarios are developed. Fifth, evaluation plans begin to take shape as early scenarios are “played out” and the expectations of the designers are made explicit.

### **Using Program Theory in the Modular Content Archives Project**

Based on the map in figure 7, the following theory of change was developed for the modular content archives project:

*Given that*

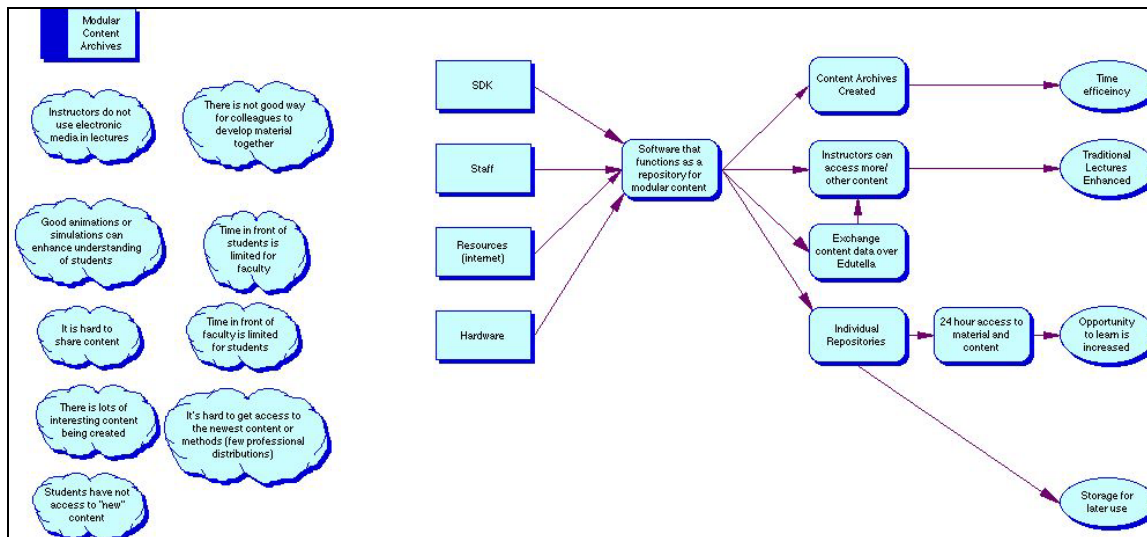
- *Instructors do not use electronic media in lectures*
- *There is no good way for colleagues to develop instructional material together*
- *Good animations and simulations can enhance the understanding students have about material*
- *It's hard to share content*
- *There's lots of interesting content being created*
- *Students do not have access to “new” content*
- *Time with students is limited for faculty*
- *Time with faculty is limited for students*
- *It's hard to get access to the newest content for teaching (few professional distributions)*

*we believe that developing software that functions as a repository for modular content will influence*

- *The creation of content archives*
- *Instructors accessing more content*
- *The exchange of content data over Edutella*
- *The creation of individual repositories*
- *24 hour access to material and content*

*which will influence the attainment of*

- *Greater time efficiency*
- *Enhanced traditional lectures*
- *Increased opportunity to learn; and*
- *Storage of content for later use.*



**Figure 4.** Program logic map for the Modular Content Archives project

*How the map was developed*

This initial program theory represented by the program logic map was elicited over the course of approximately one hour from the project's leader. A facilitator (one of the authors) stood between the project leader and a white board. As the project leader and facilitator talked, the facilitator began to diagram the project's theory of change in the form of a program logic map on the white board. In the beginning of the process the facilitator attempts to "surround" the project by first eliciting the presumed ultimate goals of the project on the right side of the white board, then the project needs on the left. This leaves a large white space in the middle of the board. In a third phase of the discussion this blank area is filled in with the intermediate goals that link the activities to the ultimate goal of the project.

### **Stating an Ultimate Goal**

Weiss (1998) notes that a useful way to begin conceptualizing a program is by looking at what it is trying to accomplish, sensibly starting with its official goals. While official goals only represent one view, they are a sensible starting place. In the case of the MoCA program logic session, the facilitator asked the project leader to state what they believe the ultimate goal of the project is.

The project leader indicated that the "ultimate goal of the project is to develop a piece of software with the aim of facilitating exchange and access of data." The project leader is very clear, in his mind, about the solution as a physical manifestation (software). At this early stage of the development of the logic map, the facilitator writes "Software" in an ellipse on the far right of the white board. As one can see from in Figure 3, the ultimate goal in the final version of the map (represented by ellipses) is not a physical, tangible artifact (such as software), but rather a pedagogically-related goal.

### **Defining Need**

Working only with the knowledge given so far about the project (that the goal of the project is to develop software), the project leader is asked why this initiative is needed and to define the problems bringing about the need. Essentially the facilitator asked: “You’ve decided to provide resources for the creation of this key bit of software that doesn’t exist today. Why? What are the problems you’ve identified that suggest you should develop this software?”

Initially the project leader found it difficult to provide an answer. In many applied-development projects, the project members tacitly hold the reasons or needs for the project; in fact they may be deemed too obvious to mention. Therefore, citing them concisely can often be a challenge for project designers. This was the case here but after some silent thought a conversation ensued and a large list of needs emerged. This list of needs is represented on the left of the map in the cloud formations and is referred to as the contextual cloud.

### **Tackling the Middle of the Map**

The project leader is asked to list the activities (sometimes referred to as “inputs”) needed in the project to reach the ultimate goal. In this case, the answer was easy for the project member. He essentially responded, “It’s software development! We will need things like a software development kit, staff...”

At this point in the process, project members in the room may have begun to wonder what was so revealing about this entire exercise. After all, on the board before them is a diagram suggesting “if one hires software developers you get software.” Revelations such as this are supposed to help with project development and evaluation? Yes, exactly. Because something deeper is really revealed at this point: after twenty minutes of discussion about a project, **the most that is known** is that if you hire software developers you get software. “Is that just it,

then?” asks the facilitator provocatively. “Your software’s not supposed to result in something good happening for student or learners? Is your software supposed to have some kind of intended effect?”

Suddenly a great deal of tacit thinking is revealed. “Oh, yes. Of course,” the project developer says. “We expect it to lead to *greater time efficiency; enhance traditional lectures; increased opportunity to learn; and provide storage of content for later use.*”

A basic gap in the project’s theory of change is revealed and the facilitator redraws the map, now looking very different from before. The ellipse “Software” is changed to a rounded rectangle to reflect its new status as an intermediate goal, and the right side of the board is populated with the new ultimate goals. The group starts to nod, realizing and approving the shift. In the facilitation of revealing a project’s program theory, we see this quite often. We refer to this as the “ultimate goal shift.” In this shift, the goal of ultimate interest is more likely to be revealed.

Lohr (1995) argues that the ultimate goal of interest must be “inherently valued.” The idea here being that the policy relevance of the project—the probability of its usefulness—will be enhanced if the outcome of interest is inherently valued. How does a project team know if they have inherently valued outcomes? Inherently valued ultimate goals are those in which there is an interest for its own sake rather than for the sake of achieving something further. For instance, according to Lohr (1995), outcome *Y* is said to be inherently valued as follows:

- If *Y* is attained, the attainment of outcomes to the right is immaterial—one does not particularly care to learn whether they occurred or even what they are.

- If *Y* is attained, one is willing to *assume* that the specified outcomes further to the right will also be attained at a satisfactory level.

With this method evaluators, system designers, managers, and clients, via discussion, select the outcome of interest to their own satisfaction. Furthermore, when the two above conditions are met, support for the project among stakeholders is significantly strengthened.

In any given project situation, solutions to perceived needs are generally articulated via the disciplinary lens of the respective stakeholders. The first thing a person usually starts to talk about to solve a problem is the implementation of well-known solutions within their field or discipline. Media people jump to media solutions. Technicians and computer scientists jump to software development. There's nothing bad about this. The key is to ask one's self if the solution proffered is the outcome of ultimate interest.

### **Linking the elements of the theory in the map**

With a "new" ultimate goal identified (while it seems new to the project members, there are those that believe it's not new at all, but just previously deemed "not this project's problem") a spirited dialog ensued wherein a set of intermediate goals now shown in the map above were linked to the new ultimate goals. Again, software is *not* the ultimate goal, but now correctly viewed as a conduit, or step, to another goal. In closing the session, a long discussion takes place on the topic of what the end ultimate goal *really is*. How far out to the right should the believed influence of the software be shown?

### ***General issues the theory of change reveals***

While there are areas where the articulation of this project can be improved through refined iterations of a program theory approach, several important things are revealed by this initial pass. First, instances where the principal investigator is very clearly set on creating a physical artifact as the solution to the problem are revealed. In this case, the physical artifact is a piece of

software. The mere creation, and therefore empirical existence, of the software constituted a solution in the early minds of the project participants. Second, an initially stated ultimate goal may not be the real (or even relevant) ultimate goal. Third, it may be revealed that the rationale for many attractive sounding projects eludes even the lead project members. Often the answers regarding “rationale for the project” are unclear. We find the reason for this is because the rationales are assumed or tacit. Fourth, that a project is believed to solve many more problems that it actually can solve may also be revealed.

***New questions about the project revealed by program theory***

The resultant “ultimate goal shift” in the program theory articulation paints a picture of the project that uncovers new questions that heretofore were not considered:

1. How will one know the ultimate goals, as mapped, when one sees them? What compelling evidence will exist to prove “Time Efficiency” has been created, that “Traditional Lectures” have really been enhanced” that “Opportunity to Learn” really increased and “Storage for Later” has been defined in such a way that it can be measured?
2. How strong are the connections between the activities in the project and the ultimate goals?
3. Do the ultimate goals really represent a solution to the problems defined as the impetus of the project?

As a result of developing a program logic map and defining the project’s program theory for the MoCA project, the following information, previously unknown to the MoCA investigator, was discovered:

- The principal investigator was very clearly set on the creation of a physical artifact (the computer support) and less on the influence that artifact would have on the learning situation; a perfectly understandable position in which to be.
- Creating a physical artifact (the computer support) was not the ultimate goal of the project, as previously believed. The creation of the computer support was an intermediate goal and means to an end.
- The evidence for detecting the attainment of the ultimate goals was unspecified. Operational definitions of what the computer support will enhance were often times tacit or missing.
- It is possible that, in the initial formulation of the project need, the goals which the computer support are likely to influence do not necessarily represent solutions that meet defined need for the project.

**Limitations related to the application of both methods in this case (More than one theory makes program logic stronger)**

In spite of the rich information revealed in this case, the program logic of this project would be strengthened by having other stakeholders a) in the room providing input at the time this was developed, or b) providing their own theory of change separately and merging it with theories of change from other stakeholders. Theories of change developed with one project member are just that: the view of how a project works in the mind of one person. Adding other stakeholder theories ensures the project is designed to meet the needs of all those involved.

This also holds for the use of scenarios. The set of scenarios would be strengthened by having other stakeholder writing, reading and providing feedback on the scenarios. Until the scenarios are discussed in a larger group of stakeholders they lack evidence for their validity.

The same is true for all the pros and cons of certain features. Program theory, as well as scenario-based design, rely on different perspectives provided by the stakeholder in order to be effective tools.

A necessary next step in the program logic approach not discussed here is a requirements review, determining what the requirements are for defining and completing all the tasks suggested by the program logic map. Initiating a requirements review is a recommended strategy to avoid allowing stakeholders having a false sense of goal alignment. By engaging in a requirements review of the elements of the program logic map, project teams ensure there are no hidden assumptions about the elements of the initiative. Project planners must ask themselves, for every element (rectangle, rounded rectangle, and ellipse): “What is the plan for this input or activity?” The more specific the answers to this question the stronger the linkages can become.

### **C. Considering the Models Together**

Program theory and scenario-based design represent complementary views on the design and evaluation of computer supported learning environments. After a brief comparison of both approaches we will sketch ways in which the two approaches can be used in combination to form a compelling approach to improving design and evaluation of CSCL experiences. We will work out what the user can profit from using both approaches.

#### **Comparison of scenario-based design and program theory**

There are important similarities between the two approaches. Both approaches stress the importance of the social context while planning computer supported learning environments. While every scenario includes at least one human actor, the ultimate goals stated in a program theory cannot be a technical artifact.

Scenario-based design as well as program theory is meant to facilitate the communication among the stakeholders. The shared understanding of how the project will work and what the program will look like in the end are important aspects in building a project and coordinating the efforts of all contributors.

Both approaches help to discover the underlying assumptions of the project in order to discuss and test them. In order to support this function they provide 'user-friendly' tools for scientists who are beginning to think about the learning utility of their CSCL solutions.

Furthermore both approaches support the design and evaluation of computer supported learning environments. During project planning and design they offer representations of the planned process and outcome. These representations reflect the assumptions and hypothesis of the stakeholders. While the project proceeds and more and more elements of the planned program are realized the assumptions and hypothesis stated in the scenarios as well as the program theory evolve and can be tested. As a consequence of this test the program as well as the scenarios and the program theory can be accommodated to the results. Scenarios and program theory are not static artifacts; they are starting point for discussion and have to be changed when necessary.

Beside these similarities there are differences between both approaches. The differences mentioned here illustrate how the both approaches complement one another:

Outcome vs. process: Using the scenario-based design approach, project members create descriptions of the project's outcome. The descriptions show the technical system in use and the educational setting that is supported by this technology. Scenarios focus on the outcome of the entire project and provide an easy to understand and vivid description of the socio-technical system.

Program theory also considers the outcome but focuses on the elements that have to be realized in order to reach the ultimate goals. The project's logic map provides a description of the intermediate stages of the development process. program theory structures the process in order to reach the goals and achieve the outcomes.

Broad range of variables vs. key aspects: Scenarios provide a complex description of the socio-technical system. Scenarios try to reflect the complexity of reality. A broad range of variables affects the interactions within a socio-technical system and has to be taken into consideration while analyzing the scenarios. Furthermore the scenarios might give an idea how the different variables interact. In contrast, program theory concentrates attention and resources on key aspects of the project (Weiss, 1998). Program theory tries to identify the variables that are seen to be the most influential for the success of the project. The project logic map provides a framework of the hypothetical interrelations of the variables and elements.

Multiple theories vs. one shared program theory: Scenarios constitute no theory about use by themselves. In fact as a representation of assumed or observed interactions within a socio-technical system they can be analyzed or interpreted according to multiple theories. Even a scenario written from the point of view of a specific theory can be examined with another theory

in mind. Thereby scenarios live up to the fact that rarely a single theory is able to explain all processes within a socio-technical system.

Program theory on the other hand represents the assumptions about how the planned program will reach its goals set out the stakeholders. A program theory ideally reflects the shared understanding of how the most influential aspects of the program interact. This shared understanding is an important guide in coordinating the efforts of the project team. A program theory nevertheless can be grounded in multiple theories and is more of a metamodel for the project.

Concrete vs. abstract: Scenarios describe interactions within a socio-technical system on a very personal level. Scenarios therefore only constitute examples of possible situations and processes. This concreteness allows to reflect about the typical or specific characteristics of that example and to think about the consequences and side effects regarding this example.

A program theory describes the socio-technical system in a more general way. Program theory focuses on the core aspects of the program and highlights general mechanisms. When initiatives or projects are being first developed, program designers can benefit from the disciplined thinking that program logic encourages (Weiss, 1998).

At large, scenario-based design and program theory hold many similarities. The major difference between them is that program theory offers a goal oriented way to structure a project while Scenario-based design proffers an explorative approach that opens the mind to the complexity of the problem, alternatives and the diversity of theories that try to explain social and socio-technical process.

While scenario-based design highlights the divergent aspects of project planning and evaluation program theory stresses the convergent aspects. For this reason scenario-based design

and program theory represent complementary views on the design and evaluation of computer supported learning environments.

<b>Feature</b>	<b>Scenario-Based Approach</b>	<b>Program Theory</b>
Perspectives on the project	Describes the intended projects <b>outcome</b>	Structures the <b>process</b> in order to reach the goals
A way to cope with complexity	Includes a <b>broad range of variables</b>	Focuses on <b>key aspects</b>
Handling varied theoretical backgrounds	Can be analyzed or interpreted <b>according to multiple theories</b>	Provides a <b>metamodel</b> of the project
Levels of abstraction	Is <b>concrete</b> more by providing examples	Is more <b>abstract</b> and highlights general mechanisms
Antropocentricity	Both reveal the of <b>social context</b> in a project and are <b>user-oriented</b>	
Contributions to multidisciplinary	Both <b>facilitate communication</b> between people with different backgrounds	
Moving discussions from the tacit to the explicit	Both try to <b>reveal the underlying assumptions</b> of a project	
Responding to the evolutionary character of design	Both are <b>open to change</b> and evolve during the course of the project.	

**Table 1.** Summarizing and comparing the features of scenario-based design and program theory approaches.

### Using both approaches

Within a multidisciplinary project, different stakeholders predefine concepts and contribute at different stages of the project and constitute specific decisions. While learning scientists predefine a certain concept of learning they are to communicate the concept as well as decisions they made. During the stage of conceptualization and implementation computer scientists predefine the technological artifact. Scenarios help to inform other stakeholders about the technological artifact. Any stakeholder is enabled to review whether a decision does actually align with the common goals. Scenarios constitute communication at any stage of the project.

Design can be understood as a process to solve problems. Even the design of computer-supported learning environments can be seen as a problem solving process. According to Carroll (2000, p. 22) we have to cope with the following six characteristic and difficult properties of design:

- Incomplete description of the problem to be addressed
- Lack of guidance on possible design moves
- The design goal or solution state cannot be known in advance
- Trade-offs among many interdependent elements
- Reliance on a diversity of knowledge and skills
- Wide-ranging and ongoing impacts on human activity

In order to solve ill-structured problems it is not possible to plan the entire program in advance. The availability of technical options can enable completely new and unforeseen ways of use and lead to new requirements. On the other hand social developments might slow down or support the use of technical options. The evolutionary aspects of software have already been highlighted by Lehman (1980, p. 1061), who stated: “But we must be concerned with the fact that performance, capability, quality in general, cannot at present be designed and built into a program *ab initio*. Rather they are gradually achieved by evolutionary change and refinement.” According to Lehman software does not only solve given problems but is as a part of the real world a source of change and new necessities by itself.

While the project has to cope with occurring changes and arising alternatives in a flexible way, it is also important to structure the design process, track decisions and build a shared model of how the program will work.

Design covers planned as well as evolutionary aspects. Therefore it is important that the project team uses methods that support divergent thinking and methods that support convergent processes. From our point of view CSCL solutions cannot be reduced to technical systems but have to take the social system into account where the technical artifact is employed in order to facilitate learning. Furthermore some decisions made during the design and planning phase are not reversible within the project or they affect ethical issues. Because of this it is very important to anticipate possible consequences and to spend some time on thinking about possible consequences and alternative solutions instead of implementing and testing ad hoc solutions.

Program theory helps to integrate each scenario, decision, and predefinition into the whole process. Scenarios force users not just to use terms but give meaningful descriptions. How do they actually want to instantiate an abstract theory of learning and teaching? This helps to implement the project within real situations of use, which are complex and ill structured. Program theory helps to focus on core aspects of design and prevent getting 'lost in scenarios'.

Scenarios and project logic maps can be used in an alternating way. Thereby it gets possible to use both approaches and improve the overall development process.

The project logic map can be a starting point for writing scenarios. Especially the interrelations between the goals and interrelation between ultimate goal and inputs can be described with a scenario. The scenario can help to understand how this interrelation is meant to work and how it will look like in a concrete situation.

Scenarios on the other hand can be used to create program logic maps by pointing out main elements of the intended program. They can also be used to completing already existing program logic maps by presenting alternative situations of use.

## **D. Conclusion**

This chapter addressed theoretical frameworks of the evaluation of learning technologies. We examined scenario-based design and program theory in the design and evaluation of computer support for collaborative learning (CSCL) and then suggested ways in which the two approaches could be used to form a compelling approach to improving design and evaluation of CSCL experiences. In sum, scenario-based design and program theory hold many similarities. The major difference between them is that program theory offers a goal oriented way to structure a project while scenario-based design provides an explorative approach that opens the mind to the complexity of the problem, alternatives and the diversity of theories that try to explain social and socio-technical process.

Scenario-based design highlights the divergent aspects of project planning and evaluation program theory stresses the convergent aspects. For CSCL researchers and developers, scenario-based design and program theory represent complementary approaches which when used together or separately, can add strength to the implementation and success of CSCL projects.

## References

- Baumgarnter, P. (1997). "Didaktische Anforderungen an (multimediale) Lernsoftware". In: L.J. Issing; P. Klimsa (Hrsg.). Information und Lernen mit Multimedia. 2., überarb. Aufl.- Weinheim: PVU, S. 240-252.
- Benner, K.M., Feather, M.S., Johnson, W.L., Zorman, L.A. (1993). "Utilizing Scenarios in the Software Development Process." In N. Prakash, C. Rolland, & B. Pernici (Eds.). Information System Development Process. Elsevier Science Publishers (pp. 117-134).
- Bødker, S. & Christiansen, E. (1997). Scenarios as springboards in design. In G. Bowker, L. Gasser, S.L. Star, & W. Turner (eds.), Social science research, technical systems and cooperative work. Erlbaum, pp. 217-234.
- Bødker, S., Christiansen, E., Thüring, M. (1995). A conceptual toolbox for designing CSCW applications. COOP '95 International Workshop on the Design of Cooperative Systems, pp. 266-284, Juan-les-Pins, January 1995.
- Breitman, K., Leite, J. (2001). Requirements Elicitation through Scenarios — a hands-on tutorial. Tutorial on the Fifth IEEE International Symposium on Requirements Engineering. Toronto. [WWW Document]. <http://www.re01.org/program.html#Tutorials>
- Carroll, J.M. (2000). Making use: scenario-based design of human -computer interactions. Cambridge: MIT press.
- Chen, H.T. & Rossi, P. (1989). Issues in the Theory-Driven Perspective, Evaluation and Program Planning, 12, pp. 299-306.
- Connell, J.P. & Kubisch, A. (1995). "Applying a Theory of Change Approach to the Evaluation of Comprehensive Community Initiatives: Progress, Prospects, and Problems." In

New Approaches to Evaluating Community Initiatives Volume 2 Theory, Measurement, and Analysis. ed. Karen Fulbright-Anderson et al. Washington, DC: Aspen Institute.

Janneck, M. (2002). Der Szenarienansatz in WISSPRO. Gekürzte Fassung. [WWW Document]. <http://mind.wisspro.de/materialdateien/szenarien-in-wisspro.pdf>

Koch, M.; Reiterer, H.; Tjoa, A.M. (1991). Software-Ergonomie: Gestaltung von EDV-Systemen; Kriterien, Methoden und Werkzeuge. Wien: Springer.

Lehman, M. M. (1980). Program, Life Cycles, and Laws of Software Evolution. Proceedings of the IEEE. Vol 68, No. 9, pp. 1060-1076.

Lohr, L. (1995). Impact analysis for program evaluation. Thousand Oaks: Sage (p. 19)

Leite, J.C.S.P.; Hadad, G.D.S.; Doorn, J.H.; Kaplan, G.N. (2000). A Scenario Construction Process. Requirements Engineering. Vol. 5. pp. 38-61.

Nash, J. B., Plugge, L., & Eurlings, A. (2001). Defining and Evaluating CSCL Evaluations. In A. Eurlings & P. Dillenbourg (Eds.), Proceedings of the European Conference on Computer-Supported Collaborative Learning (pp. 120-128). Maastricht, Netherlands: Universiteit Maastricht.

Nejdl W. (2001). EDUTELLA: A P2P Networking Infrastructure Based on RDF. <http://edutella.jxta.org/reports/edutellawhitepaper.pdf>.

Nygaard, K. (1986). Program Development as a Social Activity, INFORMATION PROCESSING 86, H.-J. Kugler (ed.), Elsevier Science Publishers B.V. (North Holland), IFIP, 1986. Proceedings from the IFIP 10th World Computer Congress, Dublin, Ireland, September 1-5, 1986), pp. 189-198.

PADLR (2001). Personalized Access to Distributed Learning Repositories – PADLR – Final Proposal. [WWW Document]. <http://projekte.learninglab.uni-hannover.de/bscw/bscw.cgi/d710/>.

Painter, M (2002). PADLR submodule: Modular Content Archives, First Technical Report. Braunschweig [WWW Document]. <http://padlr.kbs.uni-hannover.de/bscw/bscw.cgi/d7164/>.

Rolland, C.; Achour, C.B.; Cauvet, C.; Ralyté, J.; Sutcliffe, A.; Maiden, N.A.M.; Jarke, M.; Haumer, P.; Pohl, K.; Dubois, E.; Heymans, P. (1996). A Proposal for a Scenario Classification Framework. CREWS Report 96-01.

Rosson, M.B.; Carroll, J.M. (2002). Usability Engineering: Scenario-based Development of Human-Computer Interaction. San Francisco: Morgan Kaufmann.

Rumbaugh, J.; Jacobson, I; Booch, G. (1999). The Unified Modeling Language Reference Manual. Reading, Massachusetts: Addison Wesley.

Strömdahl, H., & Langerth-Zetterman, M. (2000). On Theory-anchored Evaluation Research of Educational Settings Especially Those Supported By Information and Communication Technologies (ICT) (Draft). Uppsala, Sweden: Swedish Learning Lab.

Weiss, C. (1972). Evaluation Research: Methods of Assessing Program Effectiveness. Englewood Cliffs, NJ: Prentice-Hall.

Weiss, C.H. (1995). "Nothing as Practical as Good Theory: Exploring Theory-based Evaluation for Comprehensive Community Initiatives for Children and Families." In New Approaches to Evaluating Community Initiatives: Concepts, Methods, and Contexts, ed. James Connell et al. Washington, DC: Aspen Institute.

WGLN (2002). WGLN Project Implementation Overview: Summary Analysis of BIDS Responses. Technical Report. March 21, 2002.

Wottawa, H.; Thierau, H. (1998). Lehrbuch Evaluation. 2., vollst. Überarb. Aufl. Bern: Huber.