Research and Practice in Technology Enhanced Learning
Vol. 4, No. 2 (2009) 169–189
© World Scientific Publishing Company &
Asia-Pacific Society for Computers in Education

WISE TECHNOLOGY LESSONS: MOVING FROM A LOCAL PROPRIETARY SYSTEM TO A GLOBAL OPEN SOURCE FRAMEWORK

JAMES D. SLOTTA
Canada Research Chair in Education and Technology
Ontario Institute for Studies in Education
University of Toronto, 252 Bloor St. West #11-268
Toronto, ON, M5S 1V6, Canada
jslotta@oise.utoronto.ca

TURADG ALEAHMAD*
Doctoral Student, Human Computer Interaction Institute
Carnegie Mellon University, 5000 Forbes Avenue
Pittsburgh, PA 15213, USA
turadg@cmu.edu

WISE is a software platform that enables easy authoring and exchange of educational materials amongst learning science researchers. This paper charts the recent evolution of the WISE technology, sharing lessons learned and describing a new open source framework called SAIL. We identify key challenges and design principles for technology-enhanced learning environments, and describe how we applied these principles to help WISE move from a single-developer codebase to an international open source codebase shared among multiple technology projects. Finally, we discuss the new types of community this transition fosters.

Keywords: Software framework; case study; open source; online community.

1. Introduction

This paper describes the efforts of educational researchers, technology specialists and computer scientists to produce a third-generation[1] technology platform for educational research and development. Our goal is to provide an open architecture for learning scientists, enabling the development of richly interactive materials that can be shared and co-developed between various development projects. These materials

*Author was a staff technologist in the WISE lab at the University of California, Berkeley when working on this project.
[1]The third generation refers to a sequence of technology frameworks developed by researchers at the University of California, Berkeley. The first generation was the Knowledge Integration Environment (http://kie.berkeley.edu) and the second was the closed source Web-based Inquiry Science Environment (http://wise.berkeley.edu).

should interoperate — allowing innovations developed by one research group to be easily integrated into the materials developed within another lab. Moreover, the materials should be robust in terms of the platform that serves them and stores resulting user data. In this way, we hope to sustain a dynamic open source community of designers, developers and educational researchers as they adopt and adapt materials for their own investigations, enabling collaborations between labs, and promoting the evolution of our innovations.

Our search for a scalable framework is paralleled by an explosion of progress within the areas of e-learning, digital libraries, and learning content management that has emphasized the issues of content reusability, semantic metadata, and new standards and specifications (e.g. SCORM, IEEE LOM, and IMS LD). These rapidly developing areas have cut across traditional lines of industry, government and academia, promoting dynamic communities of practice and exchange. Moreover, information technology itself has continued its rapid evolution, with new functionality in the Internet itself, in networking protocols (e.g. peer-to-peer networks, cellular devices, and wireless connectivity) and in personal and hand held computers. Our technology platforms are challenged to mature in stride with these important peer communities. This is particularly true of the e-learning community, where much intellectual effort has focused on defining technology standards and specifications (e.g. IMS, 2003; Dodds & Thropp, 2006a; 2006b; Koper, 2000) but insufficient energy has gone toward to understanding the vital issues of how technology can support new modes of learning and instruction, and new paradigms for curriculum and assessment (Friesen, 2003, 2004; Berge & Slotta, 2006).

Rather than remaining a consumer of e-learning systems, educational research must take a front seat in driving the design of new systems, posing interesting questions about the nature of learning and instruction, and investigating new forms of curriculum, new interactions between students, and new roles for the instructor. Substantial research in the design of technology enhanced learning has been conducted within programs such as the Web-based Inquiry Science Environment (WISE, http://wise.berkeley.edu), Learning Technologies in Urban Schools (LeTUS, http://www.letus.org), Thinkertools (http://thinkertools.soe.berkeley.edu/ Pages/sciwise.html), Modeling Across the Curriculum (http://mac.concord.org/), BioKids (http://www.biokids.umich.edu), and other research programs. These projects have investigated the use of models and visualizations, idea maps and modeling engines, graphs and data probes, collaboration, and inquiry learning. They have pushed the boundaries of what technology can do, providing new functionality for curriculum designs and scaffolding students, teachers and classrooms. Such research promises to transform learning from its traditional form of knowledge dissemination to new forms of knowledge community and inquiry learning (Kolodner *et al.*, 2003).

Our aim is to support education technology researchers as they continue developing such innovations, as well as to extend these innovative methods and materials to the wider learning sciences community. Graduate students working in one lab should be able to easily access the innovations from another, as well as to customize those resources for purposes of their own investigations. Two labs should be able to easily connect their learning tools and materials, to allow a greater range of functionality, as well as to promote the testing of ideas and claims (Roschelle & Kaput, 1996; Roschelle *et al.*, 1999; Clarke *et al.*, 2006). To meet this goal, a team of educational researchers, computer scientists, and technology developers from several universities have collaboratively designed and developed the Scalable Architecture for Interactive Learning (SAIL). This new open source technology framework will underlie all future tools and materials developed within our own research and that of collaborators, and will hopefully support a more global community of software and curriculum developers as well.

SAIL is concerned with the basic technology architecture of instructional environments. By carefully designing the "genetics" of digital content and functionality, we can enable the development of technology enhanced learning environments that support much more powerful kinds of learning, with more scalable, sustainable software. From the outset, SAIL has been designed to leverage the power of open source and open content development, offering a common, open architecture that will support a wide array of richly interactive learning technologies. SAIL hopes to provide a resource to an international community of designers and developers from diverse projects, leading to interoperability of technology tools, flexibility of interfaces, re-usability of content, and greater longevity and sustainability for our innovations.

Below, we describe a set of challenges that confronts all technology-enhanced learning environments. Next, we provide a case history of the WISE learning environment, in terms of how it confronted these challenges and ultimately prompted our development of SAIL. We then describe the different layers of SAIL, concluding with a discussion of how SAIL can help sustain dynamic, open source communities of developers and researchers. Finally, it should be noted that no such discussion of technology frameworks can ever be current for long. Even at the time of going to press, we are aware of new developments in the world of learning technologies (e.g. the upcoming release of Moodle 2.0 — see www.moodle.org/) to which we will be giving attention as the SAIL effort moves forward. However, we are confident that this narrative will be relevant to anyone involved in this area of work and that the arguments presented in favor of exchange communities will remain widely relevant across the learning sciences.

2. Challenges for Technology-Enhanced Learning Environments

Technology-enhanced learning environments are general systems that enable the development of technology-enhanced learning materials and instructional supports. Broader than any specific learning tool or curriculum, they address a wider class of such materials, with the aim of supporting authorship and customization of curriculum, subscription by teachers and students, and implementation within classrooms, or other educational settings. One example of such a technology environment is the Web-based Inquiry Science Environment (WISE) which enables cognitive and educational researchers to design and deliver inquiry science curriculum over the Web (Slotta, 2004; Slotta & Linn, 2009). WISE offers numerous inquiry tools such as drawing, graphing, data tables, concept mapping, online discussions and student journals to support authors who wish to create such curriculum for purposes of research or educational programs. Developed by researchers at the University of California, Berkeley, WISE has grown to serve more than 3000 teachers and 200,000 students in 35 countries and 7 languages.

While there are also many examples of so-called "learning management systems" developed for higher education (e.g. Blackboard, Moodle or Sakai), we are focused here on systems designed to support collaborative and interactive learning experience for purposes of learning science research. Another such environment designed by researchers at The Concord Consortium is called Pedagogica, and allows the construction of curriculum activities where students interact with powerful physical and biological models such as Molecular Workbench and Biologica (Horwitz & Christie, 2000). Pedagogica is not a Web-based system, and focuses on highly computational content that runs on a client-side Java platform. This fundamentally different approach to content has emerged as a direct product of the kinds of learning that was being investigated by Concord Consortium researchers. The Concord lab developed a sophisticated server-side technology to coordinate the distribution of very large application files, check versions of content via the Internet, and store and retrieve log files of detailed student interaction data. As a result of the different forms of learning that are supported by the two environments (WISE and Pedagogica), the core technologies of the two systems are quite distinct. For example, Pedagogica does not have student log-ins, databases for student interactions, or assessment tools for teachers. The authoring system is quite complex, and more of a scripting environment that requires a sophisticated knowledge of the underlying modeling environments.

Many such systems have been designed by educational researchers over the past several decades in order to support their own investigations of learning and instruction. However, the growth of e-learning in higher education and the spread of the Internet throughout all educational settings has broadened the discourse about the nature of technology infrastructures, the specification of educational content, and management of user interaction data from students and teachers. Thus, a growing community of educational researchers and developers is now concerned with the design and development of more scalable technology environments (Roschelle *et al.*, 1999; Clarke *et al.*, 2006; Wang & Hannafin, 2006; Slotta & Aleahmad, 2009). Below, we describe several fundamental challenges that confront any technology-enhanced learning environment that aims to become a sustainable technology system for research, content development and distribution to classrooms.

2.1. *Scalability*

The first challenge is that of scalability, where versions of the technology environment must be continuously and reliably extended to an increasing audience of users. Scalability is a mature area of research within computer science (Bondi, 2000; Michael *et al.*, 2007) but is a relatively new concern in education technology research (Clarke & Dede, 2009). Schools are often awkward information technology environments with various computing platforms in varying states of repair (Becker and Ravitz, 1999). While technical support may be available for the systems, it is scarce and rarely allocated to experimental pedagogy (Cuban, 2001; NRC, 2002). Drawing on more general software engineering practices, this leads to important design requirements to support education technology research.

Scaled use of any environment is difficult or impossible if the software is awkward to install or to use, or if it has substantive platform dependencies. If software bugs are fixed or new features are added, a technology environment should make such changes easy for users to update within existing software installations. Curriculum materials that run on one version of a system should continue to run on new versions. Software installation should be straightforward, and software upgrades (i.e. new versions) should be automatic and fast. These, along with other elements of sound software design (e.g. documentation, bug tracking, and support) are all essential to scalability. Additionally, there is an architectural level to this challenge, as an increasing number of users should not impact the performance of the technologies themselves. While this is clearly no problem for software that is locally installed (e.g. from a CD ROM), it may be a challenged for Web-based software, if many users impact the bandwidth of delivery. A large number of users also presents challenges for upgrading versions of software or of e-learning content, and raises challenges for support.

2.2. *Sustainability*

A second challenge is that of sustainability, which is concerned with the ability of a learning environment and its content to survive for long periods of time, in diverse contexts of use (Dalziel, 2002; Wheeler, 2004). Curricular materials originally created for purposes of research must be considered in terms of their applicability for classroom dissemination. Once disseminated, they must be able to survive within a constantly changing technology climate, with new hardware, software, networking configurations, and widely varying levels of support. For example, many curriculum projects written for the Knowledge Integration Environment (Slotta & Linn, 2000; Linn, Bell, & Davis, 2004) could never be run today, as the learning environment software is defunct, having been replaced by the WISE environment. What if the WISE software is ultimately discontinued? Would its substantial library of curricular materials be sustainable, or would they perish?

2.3. *Accessibility*

A third key challenge to technology-enhanced learning environments is that of accessibility to a wide community of users, such as other researchers, content developers, teachers and program specialists (Slotta & Linn, 2009). Is the technology easy to use and install? Can it be implemented within a classroom without overhauling the teacher's entire curriculum? Can it work well with other technologies, sharing files or other data? Can other users tailor the technology environment to their needs, adding new functionality or content? These issues are not serious when the technology environment serves mainly as a dedicated research platform. But this becomes a crucial challenge if the environment ever hopes to become scalable and sustainable. Other researchers or educators must feel that they can easily obtain, implement, and even modify the technology, as well as any content created using the technology.

2.4. *Dynamic evolution*

A related challenge is that of evolution. Most technologies tend to evolve in response to changing platforms, new features of hardware or software platforms, or indeed new approaches to teaching and learning. When Web browsers first emerged, for example, they did not have frames. Once frames became commonplace, Web-based applications regularly included them. Most researchers will improve their environments gradually over the years. For example, WISE has regularly introduced new features, and become more stable and responsive to users. However, when the changes to a system are only performed by its original authors, there is a serious risk of the status quo leading to stagnant form and function. WISE is not readily *accessible* to any developer who would want to change its interface or add new features. Thus, over the past 7 years, the WISE interface (designed to serve the purposes of WISE researchers) has remained relatively unchanged, to the point where it now is quite out-dated. As a consequence, WISE has not benefited from innovative ideas, such as the use of graphical information systems or collaborative environments, that other researchers have sought to add but could not, due to the lack of accessibility of the code. Clearly these four challenges are interrelated: only an environment that is accessible can be used by a wider community who have a greater stake in its longevity, resulting in more scalable, sustainable and dynamically evolving systems.

3. Content and Developer Communities in the WISE Environment

When learning environment technologies are accessible and scalable as defined above, they can do more than simply offer their existing content to a wider audience of teachers or researchers. Rather, the infrastructure of the environment can be opened to curriculum authors who would build new content, to technology developers who would create new functionality for the environment, and to other researchers who would adapt the environment for their own purposes (e.g. to insert their own learning tools or content into the environment, or to customize the learning environment altogether). The more open and accessible the learning environment, the greater will be the opportunities for such communities to participate and the more the learning environment will strengthen and evolve. Conversely, if a learning environment is difficult to access or to adapt, this will greatly inhibit any exchanges or dynamic evolution.

WISE has illustrated a promising degree of accessibility with regard to the development and exchange of inquiry curriculum, as content developers have employed WISE in creating curriculum that meets their professional agenda (Slotta, 2002; Slotta & Linn, 2009). For example, environmental educators working in the International Wolf Center (Ely, Minnesota) used the WISE authoring system to design and develop an inquiry-oriented curriculum concerned with wolf management (Strauss, Kerber & Slotta, 2000). This project has been run by dozens of teachers in several states, and has been translated into three different languages. More importantly, any teacher who runs the wolf project can edit the content, delete steps, add new Web pages, or change the discussion topics. Such customizations are fairly common, and to date there are more than twenty distinct version of the wolf curriculum — one of which changed the entire topic to that of salmon preservation! The Monterey Bay Aquarium (Monterey, California) has co-authored a WISE curriculum concerned with habitat adaptation of marine species (Zimmerman & Stage, 2008). The American Physiology Society has created several inquiry projects in health related topics (e.g. physiology of fitness, organic foods). The U.S. Forestry service has sponsored the design and development of a forest fire project that helps students debate whether new homes should be permitted to be constructed next to national forests.

In addition, other researcher groups have adapted WISE content for their own purposes, or authored new materials using the existing WISE learning environment. For example, researchers at the University of Tübingen (Germany) adapted the WISE Deformed Frog curriculum, translating it into German and customizing the curriculum to add "collaboration scripts" that helped students construct formal arguments (Fischer & Slotta, 2002; Kollar, Fischer, & Slotta, 2007). Micki Chi (2008) has used WISE as a platform for designing and delivering research materials for cognitive psychology research. The system allowed her to deliver this content to students in several different experimental conditions, provide them with feedback electronically, and collect all student data in real time.

Between the customizations made by teachers, the new content developed by curriculum specialists, and the adaptations made by researchers, thousands of curriculum projects have been developed over the past five or six years. None of this prodigious development activity was sponsored by any research grant, nor was it any part of the official WISE research agenda. Rather, the content development community emerged because WISE was available as an open resource to these diverse developers — a flexible system that was free of charge and provided the powerful functions that were needed to design, develop and deliver interactive content via the Web to a global community of teachers and students. Thus, WISE illustrates the promise of content development communities and indeed it has generated a great deal of excitement about possibilities in the future.

Unfortunately, WISE has not done as well in responding to the promise of open source software development. While several collaborators have used WISE as a platform for their research projects, they have been constrained to use the same version of the system as everyone else. One research group asked if they could adapt the electronic discussion tool, in order to have students within a classroom receive different discussion topics depending on what their responses had been to multiple-choice problems earlier in the WISE curriculum project. "WISE discussions can't do that," was the response given to this request, accompanied by substantial explanation concerning the architectural constraints of WISE. While the existing functionality for online discussions in WISE could have been modified with the desired functionality, our group did not have the programming staff to perform such new developments. Alternatively, the collaborators might have gotten their own programmers to attempt this change, but this would have entailed learning a great deal about the WISE software code, then modifying the existing discussion tool, resulting in two slightly different discussion modules. Inevitably came the request: "Couldn't we just give them the source code to branch out and make their own WISE?" In addition to the very real challenge of trying to develop a code base that was not designed from the outset as a collaborative code base, there was a concern that with multiple (and increasingly divergent) versions of WISE loose in the world, we might well be defeating the initial goals of our collaboration. In the end, the collaborator made do with the existing discussion tool, and hoped that one day the WISE programmers would add new functionality. Their request was added to a list of features that had been requested by an ever widening community of collaborators.

The WISE technology changed very little over its 8-year lifecycle, partly because the developers have had insufficient time and resources to implement any substantial changes (e.g. to the user interface, or available learning tools). Aside from limited developer time, however, this stagnation of form has also resulted from the status quo that is inherent in any research program: As so much hard work and careful development go into achieving a specific design, once it is finally stable there are literally years' worth of research that can be done without any major revisions. In the case of WISE, the research has been concerned with design principles for curriculum, with the nature of classroom communities using WISE, and with teacher professional development. Thus, in order to dynamically evolve over time, such environments would require the productive input from a wider community of developers. While WISE is one of the most accessible, extensible learning environments ever developed by educational researchers, it is perhaps this proximity to success that so clearly reveals its limitations. Our vision of an evolving learning environment, dynamic communities of exchange, and a common technology architecture have led us to think deeply about the core technology that would underlie our next-generation systems.

4. Designing a Scalable Architecture for Interactive Learning

In 2003, the U.S. National Science Foundation funded the TELS center (Technology Enhanced Learning in Science) to study the integration of computer-based inquiry activities within science classrooms. Building on more than a decade of prior research (Bell, Davis, & Linn, 1995; Slotta & Linn, 2000; Linn & Hsi, 2000; Horwitz & Christie, 2000), TELS sought to integrate rich models, visualizations, and probeware from the Concord Consortium (e.g. Pedagogica, Molecular Workbench and CC-Probe) into the scaffolded inquiry environment of WISE. Such functionality would support TELS researchers as they designed science inquiry projects, delivered them to classrooms, and captured student assessments. It would also serve to scaffold students and teachers as they engage with such materials in the science classroom (Linn, Husic, Slotta, & Tinker, 2008).

As described above, TELS center researchers and their collaborators had used WISE previously to author inquiry activities that employed numerous learning tools (e.g. for drawing, concept mapping, graphing, etc.) resulting in a wealth of content. However, the WISE tools were embedded within a monolithic system, making it challenging to incorporate technology innovations from other research programs (e.g. the Pedagogica or other Concord software) — which was particularly daunting given that this integration was a primary aim of the TELS center. Thus, we set out upon a fundamental re-design of the basic technology paradigm underlying WISE. Our goal was to develop a broad framework that could be used to re-create WISE with a higher level of interoperability, allowing extensions to other learning environments and opening the door to an open source development community.

In order to explore all possible development strategies, we organized several formative retreats that included technology designers, educational researchers, computer scientists, and user interface specialists.[2] Participants at these retreats deliberated various technology approaches, including Web-based applications, Web browser plug-ins, and PC-based software platforms, such as Flash and Java. These retreats also addressed major issues such as platform independence, de-centralized or distributed platforms, interoperability of software elements, and the dynamic evolution and distributed maintenance of the software systems (Linn *et al.*, 2005; 2006a).

These early design discussions examined successful technologies, including operating systems (e.g. Linux, Apple OSX), Web browsers (e.g. Mozilla), and learning management systems such as LAMS (Dalziel, 2003). We also reviewed prior efforts in educational object economy (e.g. ESCOT — see Roschelle *et al.*, 1999), and current initiatives for learning content standardization (e.g. SCORM — see Dodds & Thropp, 204a; 2004b; and IMS LD — see Dalziel, 2003; Berge & Slotta, 2006). We sought to understand common principles, successful paradigms and emerging philosophies of design and development. We also wanted to take seriously the recent activity in defining e-Learning standards and specifications (IMS, 2003; Dodds & Thropp, 2004a; 2004b), as we hoped to contribute to this dialog in the form of much richer, interactive forms of learning than are typically treated by current e-Learning systems (Dodds & Thropp, 2004a; 2004b; IMS, 2003). We articulated

[2]The retreats themselves were funded by an award from IBM to the first author, to promote the study of open source as it pertains to e-Learning systems and digital content.

a set of guiding principles that have been used throughout the past six years of SAIL development (2003–2009). These principles are discussed in the following sections.

4.1. *Develop technology in separate layers*

An important principle was the need to develop separate layers of software, enabling the re-use of lower layers while allowing greater flexibility in higher ones. Many of the most successful technology frameworks are constructed from lower level technologies that are open source and widely used. For example, the success of PHP as a programming environment for the web stands on the shoulders of the MySQL database and the Apache web server, which themselves rely the Linux operating system. The integration of these open source technologies forms the popular LAMP (Linux, Apache, MySQL, PHP "stack" that is widely used to develop Web applications (Dougherty, 2001). Some other open source projects build upon LAMP to create even higher-level frameworks such as Drupal (http://drupal.org/), a web engine for online communities, or Moodle (http://moodle.org/), a higher education learning environment. This approach allows the core technology to be jointly developed and improved by a distributed network of users. Moreover, such a separation of layers leads to a consistency of practice in developing the upper layers — a practice which lends itself to the exchange of code between software systems that have been constructed from common lower levels. The SAIL design team thus sought to define distinct layers, with the lowest being that of the basic functions common to a wide array education technology applications: the definition of a learning object, the definition of a user, and how users interact with objects, etc. By carefully defining this architectural level and differentiating it from higher levels, we sought to promote a structured, cooperative approach to the development of technology-enhanced learning environments.

We have articulated four distinct layers of technology, which are named, from the bottom up: Architectures, Frameworks, Environments, and Applications. As even computer scientists are not completely settled on the definitions of these terms (Bass *et al.*, 2003), our design group does not purport to adhere to any strict standard or definition for these terms. While the terms are inspired by current practices within software engineering, their fundamental purpose is to discriminate amongst several distinct aspects of the software and to guide our own development efforts. While SAIL is fundamentally an architecture (hence the "A" in its acronym), it must be combined with a framework layer in order to define a useful resource for any developer community. Next, we will offer definitions and examples for each of these layers, and describe how the SAIL architecture, when combined with a suitable framework layer can enable a new generation of learning environments that allow greater levels of interoperability, adaptability, and sustainability of software resources.

The *architecture* is the set of specifications for all technology elements within the overall learning environment, providing the basic set of rules within which all other elements can be constructed (Bondi, 2000; Michael *et al.*, 2007). It includes the "data model" for a learning object (i.e. what format must a learning object adhere to, and what variables or properties or meta-tags must all learning objects have), as well as a specification of a user (i.e. what is a user, how are they represented within the system, and how is their interaction with learning objects captured). This can also be thought of as the basic software commitments of the learning environment. In approaching our design of a new version of the WISE environment, we sought to greatly refine and formalize the architecture in order to guide or constrain the "permissible moves" that could be performed in designing higher layers.

The *framework* is a set of software components that extend and assemble the architectural components to a more specific purpose. This layer provides the functional affordances to the environment: Will students be able to log into it? Will teachers be able to see their students during the curriculum implementation? Can students interact with each other? Will data be saved during or after the curriculum offering? What kinds of features, materials or pedagogical structure can a curriculum activity include? What kinds of user interfaces are permitted? In the earlier version of WISE, this framework consisted of the definition of a "curriculum project" that consisted of a sequence of "steps," each of which could be of a different "type" (e.g. display an HTML page, collect a student reflection note or journal entry, provide a model or a drawing tool, or an online discussion). Other features of the WISE framework include the data structure whereby students are associated with teachers and class periods, as well as the notion of an "offering" (i.e. classroom run), and a teacher portal that included authoring permissions, assessment of student work, and online communities. SAIL enables the developer to either reuse these functional affordances or implement new ones.

The *environment* is the software system constructed according to the framework: the system of complementary applications that cooperate with each other to provide a coherent set of functions including authoring of content, storage of content, distribution of content to students and teachers, administration of content to students working in classrooms or at home, collection of data, assessment tools for teachers, and so on. If the architecture and framework are like blueprint and building materials, respectively, the environment is the building itself — albeit with no furniture or decor.

The final layer of a technology-enhanced learning environment is that of the *applications*. In the building analogy, this is the building in use: what it looks like with paint, furniture and people. The applications are what the end users (students and teachers) actually interact with: a curriculum, an authoring tool, a grading tool, etc. If the separation of layers were done carefully, the major elements of this layer would be restricted primarily to the user interface.

These four layers are not independent of one another. That is, the possible structure or implementation of one layer will be constrained by the design of underlying layers. For example, the design of the environment layer will always rely on features in the lower level framework layer, just as the framework layer has substantial requirements or constraints derived from the architectural layer. However, the layers are still clearly defined and separable from one another: It would be possible to implement several distinct frameworks based on a given architecture, or several different environments using the same framework. Indeed, as we discuss below, our own research group has created two distinct frameworks that use the SAIL architecture, in order to support a Java client-based and Web-based learning environment. Each of these layers can be created and sustained by overlapping or disjoint groups of developers, with different goals, sharing practices and software licenses (Linn *et al.*, 2006a). In essence, the different layers of technology-enhanced learning environments correspond roughly to different communities that might share and exchange source code or content within those layers (as in the developer communities of the LAMP framework). We describe the communities of SAIL and their change over time in Section 5, Experience with SAIL.

In the earlier version of the WISE software, the lower layers were not designed in such a way that they were separable from the higher ones, making it very difficult to add new applications to the environment, or to change the environment within the existing framework. In designing SAIL, we sought to separate these layers so that a new generation of learning environments could easily accommodate new features, exchange elements and interoperate with one another. After substantial design efforts, we have recognized the framework layer as best suited to foster collaborations amongst technology developers. The architecture (SAIL) may specify the structure of data and the definitions of users and groups, but the framework layer will establish the kinds of software that can be developed, the core services, data structures, and data mining capabilities. The key question is: what technologies are the most important ones for researchers to be able to share, such that they can still build the widest range of possible end-user applications? Certain features, like user registration systems and data repositories, are best if shared, since they are difficult to develop and not highly sensitive to the end user environments. We have created a powerful new Web-based framework to complement the SAIL architecture, and have employed it in developing new open source versions of WISE, although this framework is well suited to the development of many other environments.

4.2. *Allow for centralized or distributed technology frameworks*

One important characteristic of a technology-enhanced learning environment is whether it is centralized or distributed. Like most Web-based environments, WISE is *centralized*, meaning that the software is developed and released by a central authority, and the functionality and content is typically delivered via a central Web server. Like other centralized Web-based applications, WISE offers all users a single source (URL) where they log in, manage their accounts, and receive relevant services depending on their role (i.e. teachers, students and researchers all receive somewhat different functions from WISE). The WISE server collects all data that results from user interactions, such as students' work in the WISE curriculum, teachers' assessments, or authors' new versions of the curriculum.

In *distributed* technology environments, which are a relatively recent arrival in the learning sciences, users log into their own computer, and data may be saved locally or remotely at various locations. There is not necessarily even the need for a central server or authority that administers user authentications and accounts, or manages data storage.[3] Such systems include peer-to-peer networks where various kinds of files are shared within a wide network of users who connect with one another by means of mutual; negotiation (e.g. sharing their addresses) or discovery (finding other users who share membership). While no central server is required, servers can play a variety of roles in such systems.

Distributed technology frameworks emerged as a result of network capabilities to support peer-to-peer exchanges of information and software (Antoniadis & Le Grand, 2007). Napster was one of the first prominent examples of an exchange community where there was little involvement of a centralized authority (http://en.wikipedia.org/wiki/Napster). Rather, each individual "peer" who logs on is connected to all other peers in a network, which allows them to view and exchange files with any other peer. While Napster relied on a central server to establish this interconnected network of users, subsequent environments (e.g. Fastrack, Gnutella, eMule) solved this problem through *discovery* where any local peer who is using a specific kind of software is able to discover all other peers who are connected to the network and running the same peer software. This arrangement of interconnected users allows for powerful new kinds of functionality that do not require a central authority to broker transactions or store data (Schoder & Fishback, 2005; Buford, Yu, & Lua, 2008). Other examples of peer-to-peer, distributed systems are Bonjour chat in iChat (for communications) and Geo gateway (for sharing of graphical information data).

The strengths of centralized systems in general and Web-based applications in particular are undeniable. WISE has benefited greatly from the simplicity and power of this approach, and would only abandon it in the face of serious weaknesses or tremendous opportunities from a distributed paradigm. The weaknesses of a centralized approach are few, but indomitable. One important weakness is that of the single point source of failure: When the Internet service goes down, or the server crashes, all applications and content are disabled. Web-based applications also suffer from the liability of the Internet connectivity of their users. This is a particularly big problem in schools, where Internet connections and networking service is often faulty. But perhaps the greatest limitation of centralized frameworks results from the very source of their greatest strength: their single instantiation. When two or more stakeholders disagree about a change to that single instance, only one can prevail.

[3]This would be true for pure "peer-to-peer" systems, but necessarily for all distributed systems. The architecture of distributed systems should be of great importance to future research in the learning sciences.

A distributed framework can facilitate multiple autonomous technology-enhanced learning environments that cooperate, diverge and recombine as necessary.

By design, SAIL can support both centralized and distributed systems, depending on the commitments made at the *framework* layer. For instance, the TELS center will continue to develop a centralized system in order to maintain continuity with previous generations of software (i.e. WISE). However, other SAIL developers are working on a more distributed framework, with the aim of enabling new forms of interactive technology and supporting new technology configurations. We offer the term "distributed framework" to differentiate from peer-to-peer systems, which often benefit from all users adhering to a common software application (e.g. the most recent version of a certain gaming software). There have been developments in distributed frameworks (e.g. Brusilovsky & Nijhavan, 2002) and the communities for research and practice with educational technology will benefit by advancing these further. In our view, distributed frameworks must support a wide range of educational form and function, including diverse software applications, real time data acquisition, varied media formats (e.g. Flash, MPEG, HTML) and numerous specialized learning tools. Further, distributed frameworks should support different kinds of devices, such as laptop computers, cell phones, or other kinds of handheld computers. Finally, for much of the curricular content that occurs within a learning environment that is implemented on a distributed framework, we would expect a *layer* of user interaction data that will need to be stored and manipulated flexibly. SAIL provides for all this and thus facilitates these types of designs in the *applications* layer.

The potential benefits of distributed frameworks in the development of education technology are profound. By providing a clear specification of learning content, we can enable many different applications to share resources. For example, HTML is well enough specified that many different Web browsers can display any given HTML document. Moreover, by specifying different layers for the user interface and the actual content within the resource, (e.g. specific to some environment), the same resource could appear quite differently and even function differently within different learning environments (e.g. if a particular learning object is accessed via a Web browser, a cell phone, or a hand held computer — see Lee, Ko, & Fox, 2003; Eap, Gasevic, & Lin, 2009). A distributed framework can allow for much greater innovation, as end users are freed from limitations on their content, the components used within their content, and even the tools used to create it. An extensible distributed framework allows for easy integration of new and existing technologies. By distributing the naming, acquisition and authority of content and functionality, we break open the single point of innovation bottleneck and let all the ideas flow.

5. Experience with SAIL

The SAIL design team sought to carefully design and implement the four layers described above to support interactive learning content in varied forms (e.g. to be flexible to different device constraints). Rich physics and chemistry models and simulations developed by the Concord Consortium (http://www.concord.org) require a full desktop computer with the latest versions of Java. Other researchers have wanted to explore learning interactions on low-powered hand-held computers and cell phones, or even ubiquitous computing technologies that are not recognizable as computers at all (e.g. smart rooms that speak fluidly with individual computers). Our challenge was to create a software model that accommodated the diverse needs of education technology researchers, supporting with common functionality and also giving freedom in exploring uncommon designs.

The first full application of the SAIL architecture was WISE version 3.0: a reimplementation of the Web-based WISE to a Java application on the client machine that interacted with distributed web services. Several SAIL software components and services made up the *framework* level. These included the SAIL Core content framework for portable and re-mixable content, the SAIL Launching Service for delivering content and Java software components to the client, and the SAIL Data Service for decoupling data storage from content delivery and client interaction. On this, we constructed several different *environment* layers. One was Pas (for Project-activity-step), a generic version of the traditional WISE environment. Another was OTrunk, which the Concord Consortium developed as it aligned more closely with its other software components. With Pas, the TELS Center was able to create WISE *application* instances. The Pas portal component became the WISE portal, and learner activities created with Pas were WISE applications. With OTrunk, a more flexible and raw environment, Concord Consortium was able to create highly varied learner activities at the application layer.

As SAIL earned the confidence of developers and researchers, it was brought into new application contexts. Critically, it had to evolve from a technology research system to a robust technology for education research per se. The WISE 3.0 system was difficult to run in some classrooms, due to old versions of Java and some persistent challenges with platform conflicts. In addition, the WISE group was funded to develop an interactive curriculum platform for a California state-wide initiative in which the software had to run on even more impoverished computers and network. TELS researchers decided that they needed an environment where students could run content simply by clicking on a link in a web page. SAIL developers worked hard to improve the Java Web Start system by which the Java client side could be loaded with a click, but old and locked-down school computer networks proved formidable. Thus, it was decided to experiment with adapting SAIL to Web-based content. In one experiment, a researcher needed his SAIL OTrunk activity to be accessible on a Google Android G1 phone, and this was accomplished by linking the OTrunk parser with a new framework level component that transformed OTrunk into Web-based activities (Zimmerman & Stage, 2008). The flexibility of the SAIL architecture and the community of exchange among its developers made this possible.

Because it was assembled on a SAIL-based framework (i.e. a framework compatible with the SAIL architecture), many of the software components that were developed for the WISE 3 environment could be applied within a new framework, known as "WISE 4.0," that was developed for a Web-based environment. Parts of the environment and framework layers were replaced to provide for new applications, and the learning environment was moved from Java back into the web browser. Remarkably, this was done in fairly short order, as developers were able to retain the portal and user model, and many of the framework components. Changing from a Java client to Web browser environment necessitated the development of a new framework layer (e.g. the launching service was no longer necessary as Java client applications had to be converted back into Javascript applications). However, because of this layered approach to development, curricular content from WISE 3 could be easily adapted to the WISE 4 environment by software that automated the conversion.

The move from WISE 3 to WISE 4 was a major one, and served to define two distinct frameworks, and hence collaborative communities. The first is committed to a Java-based end user environment, OTrunk, with important framework components like the SAIL Data Service, Launching Service, and the OTrunk markup language. This Java framework is currently under active development by the Concord Consortium, and available for adoption by other researchers for their own Java-based environments and applications (Linn *et al.*, 2006b). The second is focused on defining a framework for Web-based environments that are well suited for running in technology-challenged settings using lighter weight components (e.g. written in Flash, or as Applets). The research group at University of California, Berkeley will continue to develop WISE 4, together with the wider community of researchers that has embraced WISE as an infrastructure. Other groups will share and co-develop the Web-based framework but develop new environment and application layers. The first author of this paper is employing the SAIL architecture and Web-based framework, but his team is developing a new environment layer that implements a flexible smart classroom environment (Slotta, in press).

In the years of software development with the SAIL architecture and layer separation, there have been two distinct frameworks developed and three distinct production applications. That the components within the model have maintained some level of interoperability serves to validate the model of layer separation for education technology research (Slotta & Aleahmad, 2009). Whereas often a technology project is born, lives and dies alone, the applications built in the SAIL model have shared parts with each other so that each lives on in new ways.

All of software code in the SAIL architecture (e.g. the portal and user registration service) as well as the two frameworks described above (the Java client and Web client) is available under the GPL open source license on public project sites (e.g. http://sail.sourceforge.net), with the goal of supporting an open community of developers. The existing collaborators who are using SAIL technologies for their research can now download their pick of SAIL-based learning environments and begin developing new features, offering those back to the community, and adopting new features or tools that are developed and offered by others. By sharing clearly defined architectural and framework layers, it is easier for tools and materials developed at the environment layer to be shared and exchanged within the community. In this way, SAIL supports the kind of functionality developed as a centralized system in WISE, but allows it to be opened to wide community of developers, to be distributed to numerous servers, to evolve as new features and user interfaces are created by the community, and thus to be sustained. Thus, SAIL offers a model that allows developers to create new learning technologies or transform existing ones with confidence that their creations will integrate smoothly and support exchanges within a community.

6. Conclusion

Because of its separation of code layers and maintenance of an open source framework, we are hopeful that a wide community of researchers from the learning sciences will capitalize on SAIL for their design of technology-enhanced learning environments. One important advantage offered by SAIL is that of stability. Not every research program has the human resources required to develop and maintain a flexible framework for the design, development and delivery of interactive curriculum materials. Because it will be used (and developed) for years to come as the central resource of the TELS Center, of WISE, and several other research groups internationally, SAIL should mature as a stable, well-supported open source code base. We are optimistic that SAIL-based learning environments will be easily adoptable and adaptable within the research community, leading to strong communities of developers that cut across the normal divisions between labs and research programs.

Another major strength of SAIL is the interoperability of SAIL-based learning objects. Any developer who builds a learning environment using the SAIL architecture will be able to adopt tools and materials from any other developer. This would allow researchers from one lab to embed innovations from another lab into their own new materials, for purposes of research collaboration, to abstract design principles, or to extend the findings to new domains or populations. Researchers could also investigate classroom settings where students use multiple SAIL-based learning environments (e.g. one for real-time data collection, a different one for online collaborative design, and a third for the use of models and visualizations). The common "DNA" of the learning objects within such environments should promote fruitful research collaborations, and ensure the efficient maintenance and development of frameworks, environments and applications that adhere to the SAIL architecture.

SAIL-based learning environments will enable more diverse curriculum designs with adaptive flow of activities and greater capabilities for student collaboration. However, in responding to limitations of past systems and embracing important movements of open source and standardization, SAIL offers a more dynamic, sustainable framework, ensuring that our innovations evolve and survive even as the technology itself continues to do the same. Adopting a more scalable, sustainable architecture will increase our own capabilities to share our innovations within the community of our peers. Ideally, other researchers will then do the same, resulting in an expansion of the overall functionality available for our innovations, and supporting the growth of a dynamic community of designers, developers, researchers and teachers.

While SAIL is a software framework, it is also a community of developers who make and employ the framework. The community itself may the primary contribution of SAIL to research in technology-enhanced learning. These shared software artifacts have brought developers from different research groups together working across and between continents to build their software together and quicken the flow of ideas and technologies within the larger education technology community. To expand the benefits of this developer community beyond the SAIL architecture, the authors have recently been making progress towards a new online community system, Educoder.org. Educoder aims to help all education technology researchers and developers learn and share for the benefit of their own projects and the field of technology-enhanced learning. There are many more lessons to be learned.

Acknowledgments

This research was supported in part by the National Science Foundation (NSF) grant number CLT 03-34199 as well as by an IBM Faculty Award to the first author. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the NSF or IBM. Reprints may be requested by e-mail from either author.

References

Antoniadis, P., & Le Grand, B. (2007). Incentives for resource sharing in self-organized communities: From economics to social psychology. Digital Information Management, 2007.

Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice* (2nd ed.). Addison-Wesley.

Becker, H. J., & Ravitz, J. L. (1999). The influence of computer and internet use on teachers' pedagogical practices and perceptions. *Journal of Research on Computing in Education 31*(4) (Summer), 356–384.

Berge, O., & Slotta, J. (2006). Learning technology standards and inquiry-based learning. In A. Koohang (Ed.), *Principles and practices of the effective use of learning objects*. Informing Science Press.

Bondi, A. B. (2000). Characteristics of scalability and their impact on performance. In *Proc. the 2nd international workshop on Software and performance* (pp. 195–203). Ottawa.

Brusilovsky, P., & Nijhavan, H. (2002). A framework for adaptive e-learning based on distributed re-usable learning activities. Association for the Advancement of Computing in Education (AACE).

Buford, J., Yu, H., & Lua, E. K. (2008). P2P Networking and Applications. ISBN 30-12374-214-5, Morgan Kaufmann.

Chi, M. T. H. (2008). Three types of conceptual change: Belief revision, mental model transformation, and categorical shift. In S. Vosniadou (Ed.), *International handbook of research on conceptual change* (pp. 61–82). New York: Erlbaum.

Clarke, J., Dede, C., Ketelhut, D. J., & Nelson, B. (2006). A design-based research strategy to promote scalability for educational innovations. *Educational Technology 46*(3), 27–36.

Clarke, J., & Dede, C. (2009). Design for scalability: A case study of the River City curriculum. *Journal of Science Education and Technology.*

Cuban, L. (2001). *Oversold and underused: Computers in the classroom.* Cambridge, MA: Harvard University Press.

Dalziel, J. (2002). Enhancing web-based learning with computer assisted assessment: Pedagogical and technical considerations. In *Proc. 5th int. computer assisted assessment conference.* Loughborough.

Dalziel, J. (2003). Implementing learning design: The learning activity management system (LAMS), ASCILITE 2003, Adelaide.

Dodds, P., & Thropp, S. E. (2004a). SCORM content aggregation model, version 1.3.1. Report of the Advanced Distributed Learning Initiative, July, 2004.

Dodds, P., & Thropp, S. E. (2004b). SCORM run-time environment, version 1.3.1. Report of the Advanced Distributed Learning Initiative, July, 2004.

Dougherty, D. (2001). *LAMP: The open source web platform* (www.onlamp.com/pub/a/onlamp/2001/01/25/lamp.html).

Eap, T. M., Gasevic, D., & Lin, F. (2009). A2, Fuhua (Oscar) Lin. Personalised mobile learning content delivery: A learner centric approach. *Int. Journal of Mobile Learning and Organisation 3*(1), 84–101.

Fischer, F., & Slotta, J. D. (2002). Online-Controversen in WISE. Wie das Internet genutzt werden kann, um naturwissenschaftliches Denken zugänglich zu machen. Computer und Unterricht *11*, 27–29.

Friesen, N. (2004). Three objections to learning objects. In R. McGreal (Ed.), *Online education using learning objects* (pp. 59–70). London: Routledge.

Friesen, N. (2003). The curriculum of the body in the age of electronic mediation [Electronic Version]. *Language and Literacy: A Canadian Educational E-journal 4*(2).

Horwitz, P., & Christie, M. A. (2000). Computer-based manipulatives for teaching scientific reasoning: An example. In M. J. Jacobson & R. B. Kozma (Eds.), *Innovations in science and mathematics education: Advanced designs for technologies of learning* (pp. 163–191). Mahwah, NJ: Lawrence Erlbaum Associates.

IMS (2003). IMS Learning Design Information Model: Version 1.0 Final Specification. From http://www.imsproject.org/learningdesign/ldv1p0/imsld_infov1p0.html.

Koedinger, K., Suthers, D., & Forbus, K. (1999). Component-based construction of a science learning space. *Int. Journal of Artificial Intelligence in Education 10*, 292–313. (Also published at http://cbl.leeds.ac.uk/ijaied/home.html).

Kollar, I., Fischer, F., & Slotta, J. D. (2005). Internal and external collaboration scripts in web-based science learning at schools. Paper submitted to the *Proc. Computer Supported Collaborative Learning (CSCL) 2005.*

Kolodner, J. L., Camp, P. J., Crismond, D., Fasse, B., Gray, J., Holbrook, J., Puntambekar, S., & Ryan, M. (2003). Problem-based learning meets case-based reasoning in the middle-school science classroom: Putting learning by design into practice. *The Journal of the Learning Sciences 12*(4), 495–547.

Koper, R. (2000). From change to renewal: Educational technology foundations of electronic learning environments. Report from the Open University of the Netherlands.

Lee, S., Ko, S., & Fox, G. (2003). Adapting content for mobile devices in heterogeneous collaboration environments. In *Proc. int. conference on wireless networks*. Jun. 23–26. Las Vegas (pp. 211–217). CSREA Press.

Linn, M. C., Slotta, J. D., Tinker, R., & Horwitz, P. (2005). Technology Enhanced Learning in Science (TELS) Annual Report submitted to the National Science Foundation for NSF funded project CLT 03-34199 (Centers for Learning and Teaching): *The Educational Accelerator Center: Technology-Enhanced Learning in Science (TELS).*

Linn, M. C., Husic, F., Slotta, J., & Tinker, R. (2006a). Technology enhanced learning in science (TELS): Research programs. *Educational Technology 46*(3), 54–68.

Linn, M. C., Lee, H. S., Tinker, R., Husic, F., & Chiu, J. L. (2006b). Inquiry learning. Teaching and assessing knowledge integration in science. *Science* (New York, N.Y.) *313*(5790), 1049–1050.

Michael, M. M., Moreira, J. E., Shiloach, D., & Wisniewski, R. W. (2007). Scale-up x scale-out: A case study using Nutch/Lucene. In *Int. parallel and distributed processing symposium. IEEE international*, *IEEE*, pp. 1–8.

National Research Council (2002). Improving learning with information technology. In G. E. Pritchard (Ed.) National Academy Press, Washington D.C.

Roschelle, J., & Kaput, J. (1996). Educational software architecture and systemic impact: The promise of component software. *Journal of Educational Computing Research 14*(3), 217–228.

Roschelle, J., DiGiano, C., Koutlis, M., Repenning, A., Phillips, J., Jackiw, N., & Suthers, D. (1999). Developing educational software components. *Computer 32*(9), 50–58. Piscataway, NJ: IEEE Computer Society.

Schoder, D., & Fischbach, K. (2005). Core concepts in Peer-to-Peer (P2P) networking. In R. Subramanian & B. Goodman (Eds.), *P2P Computing: The evolution of a disruptive technology*, Idea Group Inc., Hershey.

Slotta, J. D., & Linn, M. C. (2000). The knowledge integration environment: Helping students use the internet effectively. In M. J. Jacobson & R. Kozma (Eds.), *Learning the Sciences of the 21st Century*, 193–226. Hilldale, NJ: Lawrence Erlbaum & Associates.

Slotta, J. D., & Jorde, D. (2002). How American and Norwegian Teachers Customized the WISE Learning Environment to Improve its Value and Validity. Paper presented to the Annual Meeting of the American Educational Research Association. April 1–6, 2002. New Orleans, LA.

Slotta, J. D. (2002). Partnerships in the Web-based Inquiry Science Environment (WISE). *Cognitive Studies 9*(3), 351–361.

Slotta, J. D. (2004). The Web-based Inquiry Science Environment (WISE): Scaffolding Knowledge Integration in the Science Classroom. In M. C. Linn, P. Bell & E. Davis (Eds.), *Internet environments for science education* (pp. 203–232). LEA.

Slotta, J. D. (in press). Evolving the classrooms of the future: The interplay of pedagogy, technology and community. To appear in K. Mäkitalo-Siegl, F. Kaplan, J. Zottmann & F. Fischer (Eds.), *Classroom of the Future. Orchestrating collaborative spaces*. Rotterdam: Sense.

Slotta, J. D., & Linn, M. C. (2009). *WISE Science: Inquiry and the internet in the science classroom*. Teachers College Press.

Slotta, J. D., & Aleahmad, T. (2009). Toward a technology community for the learning sciences. Panel presentation at the biennial meeting of the Computer Supported Collaborative Learning (CSCL) conference. June 6–11. Rhodes, Greece.

Strauss, A. L., Kerber, A. K., & Slotta, J. D. (2000). Wolves in Your Backyard. WISE Interactive science activity included in the International Wolf Center's Gray Wolves, Gray Matter. From http://www.wolf.org/wolves/learn/educator/gwgm/gwgm.asp.

Wang, F., & Hannafin, M. J. (2005). Design-based research and technology-enhanced learning environments. *ETR&D 53*(4), 5–23.

Wheeler, B. 2004. Open source 2007. *EduCause Review*. July/August 2004.

Zimmerman, T. D., & Stage, E. K. (2008). Teaching science for understanding. In L. Darling-Hammond (Ed.), *Powerful learning: Teaching for understanding in the classroom* (pp. 151–191). San Francisco, CA: Jossey-Bass.