Software Development as Music Education Research

Andrew R. Brown
Queensland University of Technology

Abstract
This paper discusses how software development can be used as a method for music education research. It explains how software development can externalize ideas, stimulate action and reflection, and provide evidence to support the educative value of new software-based experiences. Parallels between the interactive software development process and established research methods are drawn, with particular focus on action research, case study, and activity theory. A new approach to arts educational research called Software Development as Research (SoDaR) is proposed. The paper includes examples from the author's use of this approach when developing the *jam2jam* software to facilitate networked music improvisation experiences for young children.

Writing computer software to enable learning is much like any educational resource design. It turns concepts about the learning experience into a concrete activity that can be applied and tested. In this paper I will describe a research approach I call Software Development as Research (SoDaR), where software is developed to enable new learning experiences. Student engagement with these experiences is assessed for educative value. The SoDaR approach involves the concurrent cyclical development of theories, activities, and software. To illustrate aspects of the SoDaR approach throughout this paper, I will discuss the development of the *jam2jam* software which was designed for computer-assisted music improvisation over a network (see Figure 1).
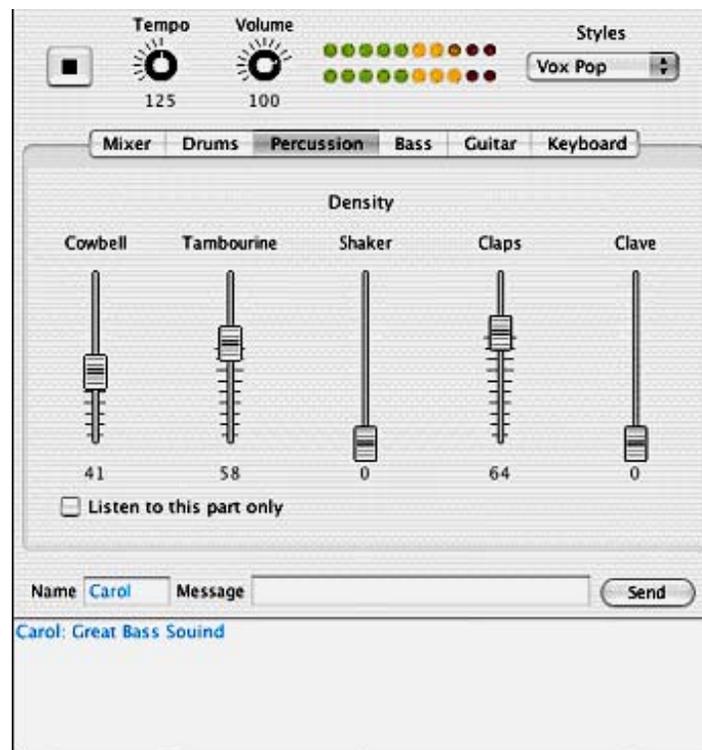

Figure 1. The *jam2jam* software.

The *jam2jam* software development started in 2002 as a project for the Delaware Children's Music Festival, held in Ohio, USA. The festival organizers were seeking an innovative music experience for children and, specifically, a project to accompany a computer installation at a local community centre and childcare facility. In response to this brief, the author and Dr. Steve Dillon proposed the idea of internet jamming to the festival organizers who agreed that we would install the software in the community center's new computer laboratory, run workshops for children there and in schools, and conduct jamming sessions for the general public during the festival. We developed and wrote the first version of the software in a few months prior to visiting Delaware in 2002 where we engaged in a

week of intensive workshops and software revisions.

The SoDaR approach draws on a number of existing research designs from the qualitative and software development domains and combines them in a unique way. The articulation of this research design is timely because software development is becoming an increasingly important mode of expression in our society, and the writing of specialized software systems for educational purposes is now well within the reach of many researchers and educators. The proliferation of software design for creative education is evident in the work of Mitchel Resnick (1994), Allison Druin (1999) and, specifically for music, in the work of Phil Ellis (1995), Jeanne Bamberger (Bamberger & Hernandez 2000), and Tod Machover and his students work on the Toy Symphony (Machover 2003).

The rest of this paper outlines a discussion of the background influences that have informed the SoDaR approach, in particular looking at established research designs and how SoDaR relates to them. Second, the outline the SoDaR approach is explicated in detail, describing three stages and the processes undertaken at each. Third a number of the issues which arise when undertaking research using the SoDaR approach are discussed and questions often asked about the research design are addreseed. Finally, the implications for reporting SoDaR research are considered.

Toward a New Research Design

The SoDaR approach has three stages; 1) identification of the learning opportunity for which software development is required and establishing an appropriate approach to take advantage of that opportunity, 2) design and production of the software, and 3) implementation and refinement of the software via application in an educational setting. Each of these three stages include processes of description, data collection, and reflection. There are some existing practices that SoDaR takes inspiration from, including software development cycle, extreme programming, action research, case study research, and activity theory. In the following sections how these influence the SoDaR approach will be examined. The software development cycle is widely used in software engineering. It focuses on the delivery of complete and robust products to the software user (Kendall & Kendall 1988). The focus on cyclical iteration has been maintained, if not reinforced, by more recent software development methodologies, most notably Extreme Programming (Wells 2001) which has listed amongst its rules "The project is divided into iterations." The steps in a software development cycle vary between each author architect, However, the elements consistently fall into these categories; Planning, Designing, Developing, and Testing. All software development cycles rely upon iteration between the making and testing stages to evolve the product to a robust state, although the extent of interaction between stages may vary between implementations. From software development processes SoDaR inherits the iterative nature of the making and testing. However, it varies from traditional software development practices in that the developmental process is exposed to real-world situations at each stage, where traditional software developers would typically be exposing the work to internal

testing regimes, or alpha testers at best. The Extreme Programming (XP) methodology is an exception to the rule, in that it encourages early real-world testing of minimally featured software prototypes. The rules of XP that indicate this tendency include "The customer is always available" and "No functionality is added early" (Wells 2001). Putting early prototypes in the field provides feedback about the struggles and novel uses of the software that can reveal both user understandings and patterns of thinking, and how well the learning theories are embodied in the software activity The SoDaR approach uses observations of intentional variations in a system to reveal the ways in which changes influence understanding. It in this way it has similarities to action research.

Action Research is focused on repeatedly observing developments in a deliberately altered situation (Cherry, 1999). Action Research often employs a cyclical process of action and observation which is a feature of the SoDaR approach in the third stage. Action Research is usually situated as a naturalistic research method (Erlaandson, et al. 1993) relying especially on qualitative data methods. Action Research is well established as an educational research strategy and the SoDaR approach shares with it the benefits of contextual and iterative intervention to improve learning situations. While this is largely true for SoDaR, the use of quantitative procedures derived from the psychologically-focused human-computer interface field may also be employed. This mixture of quantitative and qualitative methods applied to the one situation is reminiscent of case study methodology.

Case Study methods are focused on generating a rich understanding of a case by gathering data from multiple perspectives (Bromley, 1986). An important question for case study researchers is; what is the case? (Yin, 1989). The focus of investigation when using the SoDaR approach is the relationship between software and the people who use it. To achieve a greater sense of generalization when using the SoDaR approach, a multiple-site study is often undertaken. Both the technology and the participants are considered to be active in the process, therefore an understanding of both the affordances of the software and what interactive behavior reveals about human perception and understanding are of interest. Because SoDaR investigations involve interaction with technology the approach has similarities to Activity Theory.

Activity Theory is concerned with understanding technologically-mediated experiences. For activity theorists, technologies can range from language and symbolic systems to mechanical and computing artifacts. Activity Theory has a psychological heritage (Vygotsky, 1986) and is concerned with how thinking is enacted during everyday activities (Nardi 1996). SoDaR investigations are similarly concerned with understandings that become evident through interaction with technologies in the complexities of real-world situations. SoDaR also shares with Activity Theory the belief that intelligence is distributed such that knowledge and skill are enacted during interaction between people and technologies in a context, rather than being idealized mental constructs (Perkins, 1992). However, unlike most Activity Theory investigations that study the interaction with an existing systems, SoDaR is concerned with deliberately employing designed systems that embody an hypothesis about educative experiences, in order to illicit new research findings.

Defining a New Research Design

The SoDaR approach is a research tool that enables new ideas about interaction, understanding or behavior to be tested through activities using purposely-designed software that facilitate specific interactions. The approach has three stages, at each stage the researcher is encouraged to describe the objectives for that stage, collect data resulting from work undertaken at that stage, and to reflect on the outcomes, problems, and progress of the research to that point.

SoDaR Stage 1: Define the activity

In this first stage, a situation is identified in which a new software application is likely to encourage interaction leading to learning. The situation is documented, including a description of the activity and its educational potential. An initial specification for the software is developed. Implicit educational and domain assumptions and theories must be made explicit at this stage in order to enable their inclusion as part of the software specification. Expectations and hypotheses about the likely outcomes are established—not for the purposes of proving or disproving the hypothesis but as a map of obvious or known outcomes that can assist in the identification of novel or innovative practices which may emerge later.

Supporting data is collected at this stage to assist in the development of the activity. This data may include examples and analysis of previous learning strategies in the area, activities or software systems similar to those proposed in other areas, and supporting literature about practical or theoretical aspects of the activity design.

Reflective questions that will help focus and refine the proposed activity may include:

- How will the activity lead to the desired learning outcomes?
- What educational value would be provided by the software?
- Why is software the best medium for providing this experience?
- Is the activity described at an appropriate level of detail?

In this stage of developing the *jam2jam* software it was decided that a collaborative and improvisational music activity was desired. It was hoped that the computer-based experience of music making would be as authentic as possible, that is, it would provide similar experiences to music improvisation on acoustic instruments. It was known that the activity would take place in a setting where there were many computers and so there would be some requirement for them to be networked. It was also decided that a generative music system would be used to scaffold music making to enable access to a wide range of age groups and prior musical experience.

SoDaR Stage 2: Software design and production

At the second stage, the software application required to enable the activity are developed. In order to keep the research opportunities as open as possible it is important that, at this stage, software development proceeds only as far as necessary to enable the activity in

a rudimentary way so that the development investment is small enough that it can be discarded if the findings suggest a different direction is required. Further refinement and feature additions will be added in the third stage. Documentation of this stage should include any modifications to the software specification, and instructions for use.

Reflective questions at this stage are designed to maintain focus on the research objectives, so as not to be distracted by the technical details of software development, and to assist in the tracking of developments which might influence interaction during the activity. Questions at this stage may include the following:

- What data structure best supports (technically and pedagogically) the domain knowledge being represented?
- How does each software feature reinforce the design objective?
- What software platform will best enable production and deployment?
- Has the design and production process limited or expanded the educational implications?

The early prototypes of *jam2jam* focused on the core aspects of a generative music 033 work several systems engine, parameter control via sliders and dials, and the ability to net These issues were significant enough that several usability considerations including .together ease of networking, providing a range of musical styles, and internet communications musical preferences was 'udents A survey of the st .between players were underdeveloped undertaken so that appropriate musical styles could be implemented in the software, with a .view to maximizing acceptance by the target group It was encouraging to have staff report that students using jam2jam were engaging in music with more enthusiasm than previous musical activities.

SoDaR Stage 3: Usage and refinement

During the third stage, several iterations of student engagement in the activity and software revision are undertaken. This stage can be as brief or as long as the situation allows, bearing in mind that the more times the engagement-revision cycle is repeated the more likely the activity and software will be robust and effective. At this stage a description of the experience as a 'lesson plan' should be written, and notes kept for each activity session detailing actions and behavior of students, comments made by participants, observable successes and failures. This process is deliberately reminiscent of design reviews advocated by Donald Schön (1987) as an important part of reflection-in-action.

Supporting data should be collected in the form of responses to the activity and changes to the activity plan and software at each cycle. Data can include interviews with participants, video footage of students engaged with the activity, bug reports, feature requests, software version histories, and the quantitative results of any pre and/or post tests or assessments of student understanding that were undertaken.

Reflective questions at this stage should focus on the relationship between the students and the learning objectives, and the degree to which the activity and software worked as a

cohesive pair. Questions may include:

- Are the activity and software mutually reinforcing?
- What are the differences between the expected and actual behavior of the students?
- How can the software and its use be improved?
- Are the students achieving the desired learning outcomes?

During this stage of the *jam2jam* development several different cycles were involved. Firstly, the early prototypes were used with a small group of children in controlled settings. From these trials it became clear that the amount of algorithm parameters exposed to the users were excessive and a more limited range was selected, and for networked improvisation sessions where students were distant that some form of text chat facility was required. Dynamic variation from note to note was included to add interest and life to the music.

As we became engaged in the details of the software, generalizations across the instrument parts became evident. This lead to changes such as controls being grouped under two consistent headings (density and parameters) which provided better internal consistency that assisted student understanding.

Secondly, an intensive period of trials and development took place during a festival. At this time user testing was much more intense and users were much more diverse in age and experience. As well as numerous small bugs being caught in this process, educational issues arose around the words used to describe features on the interface, the need for a simplified method of networking computers on a local area network, and the requirement to have improvisations recorded so that students could take away a record of their music making to show their families and friends. The social recognition and pride in achievement facilitated by the sharing of recorded outcomes reinforced important outcomes of the music festival.

Implementation Issues in Educational Settings

 SoDaR stages. Through experience in a number of projects using the method these issues have proven to be important in the effective use of the SoDaR approach.Key issues include skills and teams, adding value through software, deigning interaction, the usage context and exceeding expectations.

Skills and teams

Research is often a solo venture, where the researcher undertakes all tasks from literature review, to data collection and analysis, to presentation of findings. In educational research this is reinforced by the culture of 'solo' teacher. As a result, the software engineering skills required by SoDaR may seem a prohibitive boundary to many researchers. An obvious, and worthwhile, solution is to work in a team with a software developer. Research teams have many benefits and, in the mixed skill environment of educational institutions, locating a colleague with software development skills may not be too difficult. In relation to education, the value of computer programming to children's development was pointed out by Seymour Papert (1980), however, its value to teachers for the building of

educational materials is just as powerful. It is worth emphasizing the benefits of developing software engineering skills for those so inclined. Learning to program a computer changes the relationship between the person and the computer, such that the person can create what they need rather than having to accept only what features the machine provides. Also, as a mode of expression, computer languages provide both an alternative mode of thinking and opportunity to communicate ideas. These advantages apply to software development as they do to developing skills in other modes of expression, such as water-color painting, calculus, rhyming couplets, or music notation.

The *jam2jam* team which began with Dillon and Brown (2003) provided sufficient software engineering, research and educational expertise for the first iteration of the software and testing in the Delaware situation. This was partially possible because of the use of library code developed by Andrew Sorensen who later came into the team in a more active capacity at times when the sophistication of the software system demanded. This collaborative approach made possible the research and the development of new educational software which none of the participants would have imagined, let alone produced, on their own.

Adding value through software

The success of the SoDaR approach relies on enhancing learning activities by the addition of new software applications. Therefore, it is important that researchers understand the operation and potential of the computer as a medium for expressing ideas and building activities. On the one hand, the computer can act as a meta-medium to simulate previous media. For example, software can: act like paper for writing words or music notation, act like radio and play back audio recordings, act like TV or cinema and replay movies, and act like an audio-video edit suite and enable the manipulation of sound and moving image. When simulating existing media the computer takes on the educative value of that media which is moderated or expanded to some degree based upon differences between the traditional medium and the computer simulation in terms of cost, convenience, features and perceived relevance. In some cases the computer is a cheaper or more convenient medium, in other cases it is neater and more editable, but in a number of cases it is more expensive and more restrictive. On the other hand, the computer may be used to create new learning opportunities unlike those possible with previous media. The value of software for education is most clearly evident when the computer opens up new opportunities for music making not previously possible or accessible. Software adds greatest value when it makes a qualitative difference. For example, in the *jam2jam* software the combination of generative music algorithms and Internet connectivity were two aspects of computing technology that produced a unique learning experience for the students. The generative systems allowed students to be meta-creators controlling numerous musical attributes with one gesture (not unlike a conductor), and the network connectivity meant that real-time musical collaboration could occur between users in different locations.

Designing Interaction

At all stages of the SoDaR processes there is a degree of iteration in the activities. The reflective questions assist in this, as do the evolution of ideas and outcomes enabled by various data collection strategies. Development and refinement through iterative steps is a common practice in design situations. Because of this, the initially provisional nature of the software is welcomed in this methodology. SoDaR projects are assisted by opportunities to shift direction or change emphasis in the software as the research proceeds. This is at variance with some software engineering practices which strive to deliver complete functionality at the time of public release. Yet, it resonates with more recent participatory design processes and the practical evolution of commercial software across several generations as it varies in response to user feedback.

The SoDaR process of improvising with the software design and with its potential educational uses generates a richness of data that is most valuable to the researcher. The researcher benefits by 'playing' with the possibilities presented by the malleable nature of software, in the same way that the student learns by playing within the experiences created by the software/activity combination.

When proceeding through the SoDaR stages the speed of the iterative cycle slows down. In the activity identification stage, reflective cycles occur in the researcher's mind or in discussion; reflection and action are deeply intertwined. At the production stage, iterations in the design need to be built and explored before decisions about maintaining features are made. At the implementation stage, one or more trial sessions need to be undertaken in each reflective cycle, and revision of the software may also be necessary but significant changes can be time consuming. It is at this final stage that the iterative processes are most similar to those larger scale cycles in Action Research.

The usage context

It may be possible to assess the likely value of some computer music software prior to seeing it used by students, but any such assessment will be provisional, at best, because the potential of software is different in each educational situation in which it is used. The SoDaR approach relies upon a strong link between the software and a learning activity, recognizing that the activity creates a context of usage that will hopefully be reinforced by the software. Research findings generated using the SoDaR approach are limited to the studied contexts in the same way as other methods used within qualitative research studies. While this can be alleviated somewhat by studying multiple sites or classes, generalizations should be made with caution and context dependency taken into account. Contextual considerations beyond the software and activity include student readiness for engaging with the learning objectives, ease of physical computer access, clarity of usage objectives in the student's mind, opportunities for outcomes to be shared, and connections and links made with other activities in the student's life.

The *jam2jam* software underwent rapid developmental changes in the Delaware experience as the particularities of that situation revealed themselves. Since then the software has undergone even further changes due to different experiences and use in different

countries and with different age groups. However, the pace of development and change has slowed considerably, indicating that the bulk of the findings about learning through the *jam2jam* software were uncovered in the first few iterations and trials.

Exceeding expectations

An important SoDaR feature should be the maintenance of a healthy skepticism about the degree of control the researcher has over outcomes. In this way control is understood as improvisation rather than direction and there should be a view to generating knowledge by capturing the opportunities that arise - this in the spirit of grounded theory (Strauss 1990). One way to assist this emergent approach is by reflecting upon which elements of the process were expected or anticipated and which were surprising. Identifying the valuable opportunities from the distractions requires that a focus be maintained on the educational objectives of the activity. Noticing opportunities does not always imply adding features to the software and software developers are rightfully very wary of 'feature creep.' The educational value of software is often enhanced as much by focusing on the core value of the activity and reducing complexity more than by increasing complexity.

During student activity sessions with *jam2jam* new ideas about what was valuable about the activity frequently emerged For example, in the *3jam2jam* sessions with young children we were surprised by the attention students placed on the text messaging; given that is was a music exercise. There engagement in the text messaging was evident in their questions about how to spell words or where characters were on the computer keyboard. In the SoDaR approach the researcher needs to remain alert to these new opportunities for learning, and action-reflection cycle encouraged in SoDaR provides opportunities for accidental occurrences to be integrated into future activities.

Presentation of Research Results

 Having just described research using the SoDaR approach, there is still the questions of how such research should be presented and reported. This section will include discussions about research presentation as it applies to reporting findings in scholarly journals, conferences, and considerations for thesis preparation and assessment.

Reporting structures for SoDaR projects naturally lend themselves to a narrative focus that follows the progress through each stage. This narrative approach is similar to those of ethnographic or anthropological research and much of the literature on these methods is relevant here also, for example LeCompte & Schensul (1999). The rich media nature of data collected as a result of this approach imposes a presentational challenge in paper-based research forums. Online journals and conference presentations, where a variety of media can be used, are likely to enable a richer picture of the research to be reported. Demonstrations of the software are clearly valuable when describing the research, and video footage of students using the software is likely to be the most powerful experience for an audience. In general, examples of computer code will not be useful to most educational audiences even though

they tell a significant part of the story to the knowledgable reader.

The use of software source code examples is, however, likely to be a significant part of examinable thesis presentations of work done using this approach. The reader will be expected to have some background in software development in order to make judgments about implementation and the contribution to new knowledge, especially at the research higher degree level. SoDaR processes benefit from a strong underpinning in learning theories, user interaction, and the philosophy of technology because the process is grounded in the testing of theory through manifestation as software-based experiences. A familiarity with relevant theories can be demonstrated in literature reviews and discussions of results..

Audio visual materials on CD, DVD, or web sites can be valuable elements of the submitted work. The balance between these written, audio-visual, and software components could vary considerably from case to case depending upon the focus of the study and expertise of the researcher. SoDaR shares many of the uncertainties of formal research assessment that plague creative practice as research given that the software design is highly influenced by the personal experience and skills of the researcher, however, because the quality of the student experience is the focus, rather than the quality of the software development practice, these issues may not be as severe for theses using the SoDaR method.

Conclusion

Overall the evidence presented here suggests that software development as research (SoDaR) provides a rich opportunity for the educational researcher. Using the SoDaR approach the research processes are focused learning experiences that leverage new opportunities provided by new software-based activities.

Software development can be a useful research method because it involves the externalization of domain and learning theories and assumptions, and makes them available for experimentation and reflection. In this way software acts as a mirror on researcher understanding, an embodiment of the learning theories, and a facilitator of domain activities. The SoDaR approach uses software development to create technology mediated experiences for the student, in a process best described as building new learning environments.

The SoDaR approach involves computers, but is about people. It utilizes software engineering processes, but situates them in a context of qualitative and quantitative data collection, and encourages active and regular reflection upon the learning experience. This paper shows how software development can actively be employed as a research method and as an amplifier of domain understanding.

References

Bamberger, J. & Hernandez, A. (2000). *Impromptu* [software application]. New York, NY: Oxford University Press.

Bromley, D. B. (1986). *The case-study method in psychology and related disciplines*. London: John Wiley & Sons Ltd.

Brown, A. R., Dillon, S. & Sorensen A. (2003). *jam2jam*. Exploding Art Music Productions. http://explodingart.com.au

Cherry, N. (1999). *Action research: A pathway to action, knowledge and learning*. Melbourne : RMIT Publishing.

Druin, A. (1999). *The design of children's technology.* San Francisco: Morgan Kaufmann.

Ellis, P. (1995). "Designing sound: Developing computer support for creativity in music education" *Research Studies in Music Education*, Vol. 5:11-23.

Erlaandson, D. A., Harris E. L., Skipper, B. L. & Allen, S. D. (1993). *Doing naturalistic inquiry: A guide to methods*. London: Sage Publications.

Guba, E. G. & Lincoln, Y. S. (1989). *Fourth generation evaluation*. Newbury Park, CA: Sage Publications

Hunt, A. & Thomas, D. (2000). *The Pragmatic programmer: From journeyman to master*. Reading, MA: Addison-Wesley.

Kendall, K. & Kendall, J. (1988). *Systems analysis and design*. Englewood Cliffs, NJ: Prentice-Hall International Editions.

Kernighan, B. W. & Pike, R. (1999). *The practice of programming*. Reading, MA: Addison-Wesley.

LeCompte, M. D. & Schensul, J. J. (1999). *Designing and conducting ethnographic research*. Walnut Creek, CA: AltaMira Press.

Machover, T. (2003). *Toy Symphony*. http://www.toysymphony.net/

Nardi, B. (Ed.) (1996). *Context and Consciousness: Activity theory and human-computer interaction*. Cambridge, MA: MIT Press.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Perkins, D. (1992). *Smart Schools: From training memories to educating minds*. New York: The Free Press.

Resnick, M. (1994). *Turtles, Termites, and Traffic Jams: Explorations in massively parallel microworlds*. Cambridge, MA: MIT Press.

Schön, D. A. (1987). *Educating the reflective practitioner*. San Francisco: Jossey-Bass Inc.

Strauss, A. L. (1990). *Basics of Qualitative Research: Grounded theory procedures and techniques*. Newbury Park, CA: SAGE Publications.

Wells, D. (2001). *Extreme Programming; A gentle introduction*. Retrieved 16 September, 2004, from http://www.extremeprogramming.org/

Vygotsky, L. S. (1986). *Thought and language*. Cambridge, MA: MIT Press.

Yin, R. (1989). *Case Study Research: Design and method*. Newbury Park, CA: SAGE Publications.

About the Author

Andrew R. Brown is Associate Professor in Music and Sound at the Queensland University of Technology and the Digital Media Program Manager for the Australasian CRC for Interaction Design (ACID). Dr. Brown's expertise is in technologies that support creativity, algorithmic music and art, and the philosophy of technology. His current research focuses on adaptive music for computational arts and interactive entertainment. He is an active composer of computer music and a builder of software tools for dynamic content creation.

International Journal of Education & the Arts