Forensic Reasoning About Paleoclimatology

Kenneth M. Anderson KENA@CS.COLORADO.EDU
Elizabeth Bradley LIZB@CS.COLORADO.EDU
Laura Rassbach de Vesine LAURA.RASSBACH@COLORADO.EDU
Department of Computer Science, University of Colorado

Marek Zreda MAREK@HWR.ARIZONA.EDU
Chris Zweck CZWECK@HWR.ARIZONA.EDU
Department of Hydrology, University of Arizona

Abstract

Human experts in many scientific fields routinely work with data that are heterogeneous, noisy, and/or uncertain, as well as heuristics that are unproven and possible conclusions that are contradictory. We present a deployed software system for cosmogenic isotope dating, a domain that is fraught with these difficult issues. This system, which is called ACE ("age calculation engine"), takes as inputs the nuclide densities in a set of rock samples taken from a landform. It answers the scientific question "What geological processes could have produced this distribution of nuclide concentrations, and over what time scales?" To do this, it employs an encoded knowledge base of the possible processes that may have acted on that landform in the past, complete with the mathematics of how those processes can affect samples, and it uses a workflow system to encode the computations associated with this scientific analysis. Flexibility and extensibility were critical issues in ACE's design, in order to allow its scientist-users to modify and extend it after its developers were no longer involved. The success of this is evident; the system remains in active use to this day, several years after the development cycle ended, without a single request for help from the geochemists to the computer science side of the team. The ACE project website has received over 17,000 hits since 2008, including 2500 over the last twelve months. The software ($\sim$20,000 lines of python code) has been downloaded nearly 600 times as of April 2013, which is a significant number in a research community of $O(10^2)$ PI-level scientists.

Keywords: Scientific discovery, discovery informatics, cosmogenic isotope dating, argumentation, scientific workflow, extensibility.

1. Introduction

Science is becoming increasingly challenged by complex reasoning about noisy, heterogeneous data—often too much data, but sometimes not enough. Helping scientists manage those data, and make sense of them, are important challenges for computer science in general and discovery informatics in particular. This paper is about the lessons learned during an interdisciplinary collaboration between geoscientists who date landforms and computer scientists who spent five years creating a software tool called ACE that streamlined and enhanced that geological analysis process.

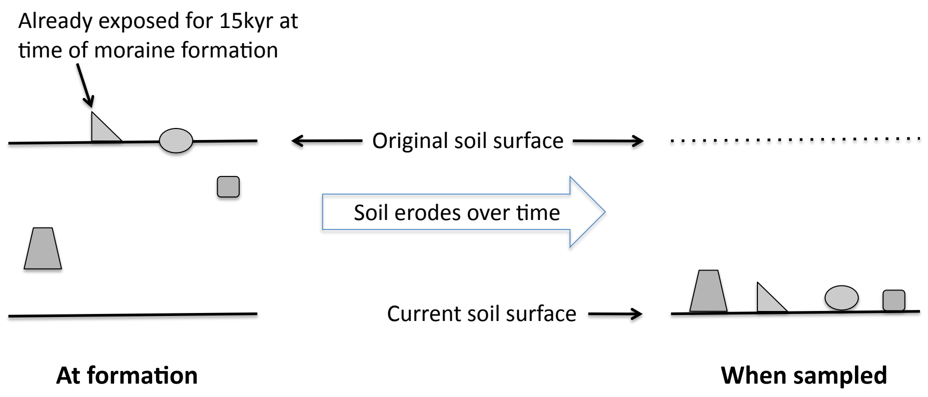


Figure 1. The evolution of a moraine

Dating landforms is very much like investigating a crime scene: from the information that is available on the surface today, experts must deduce what happened in the past. Many landforms are created by a single geological event that happens almost instantaneously in geological time. They then evolve in time in ways that are known, at least in general. Terminal moraines, for example, are formed when a glacier recedes. As these landforms age, subsurface rocks are exposed as the fine matrix around them erodes. See Figure 1 for a diagram. A geoscientist sees the situation shown on the right. If she wants to know when that moraine was formed, she needs to reason backwards to determine the initial state of the system—the situation on the left—in order to deduce the timeline. This entails figuring out what processes were involved in the evolution of the landform. Geoscientists tackle that problem by making some assumptions about those processes, projecting those assumptions backwards through time and space to the putative formation time of the landform, and iterating the process until the until the modeling results are consistent with the observations.

ACE was designed to assist expert geoscientists in carrying out the complex data-analysis and scientific reasoning task sketched in the previous paragraph. Working from a collection of exposure ages derived via radioisotope dating from rock samples, its first task was to "calibrate" the isotope dating method. That entailed determining a production rate for a particular nuclide from a set of rock samples of known ages—the "calibration set." It then used that production rate to deduce the ages of the new, undated samples. ACE incorporated a number of different calibration sets—for different cosmogenic nuclides and different scientific situations. It used a workflow engine to make it easy to create, edit, run, and evaluate new cosmogenic dating algorithms. ACE's user created a new *experiment*—the name for the basic software construct that captured all of the information about a particular run—specified a nuclide of interest for that experiment, specified a calibration set for that nuclide, and then ran the calibration workflow in the workflow engine to create the age estimates for the samples. Then, using an encoded knowledge base of rules about mathematical geoscience, ACE's reasoning engine worked through scenarios about what processes could have produced that set of sample ages. Finally, ACE reported the possible scenarios to its scientist-user, together with a narration of its reasoning about each one.

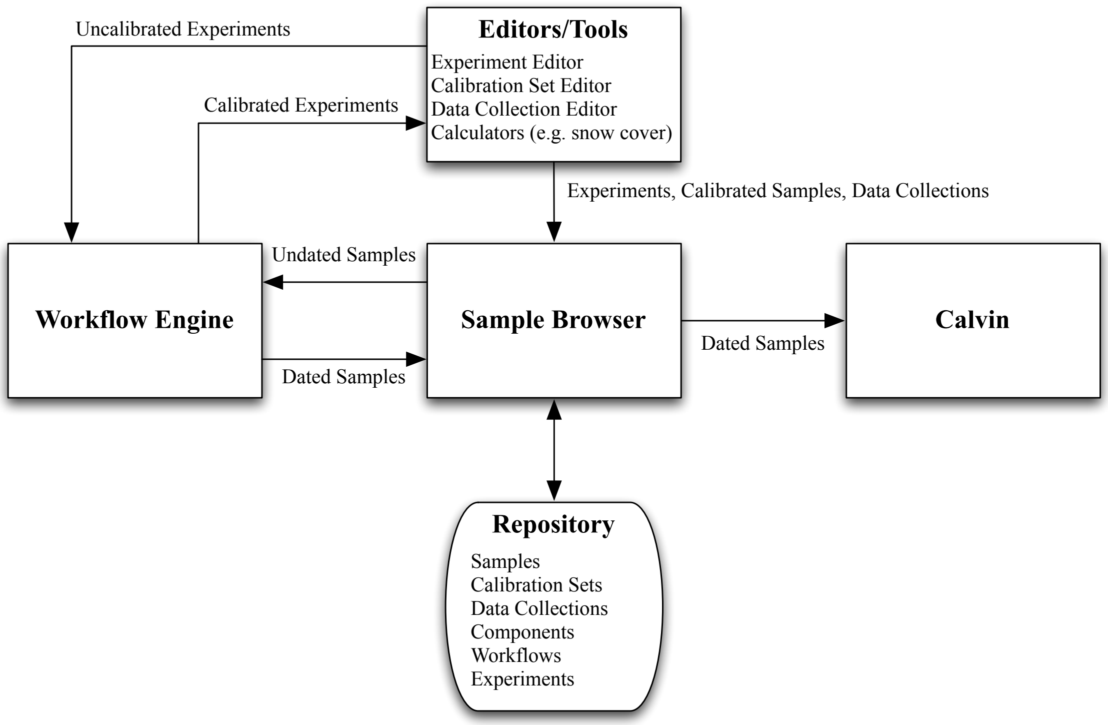


Figure 2. ACE Software Architecture

The challenges that arose during this project ranged from the obvious, such as learning each others' language, to the subtle. The notion that a computer can only do what it is programmed to do, for instance, goes without saying to a computer scientist, but is not at all obvious to those outside the field. That kind of implicit disconnect can easily stymie multidisciplinary research—or even completely derail it. And the traditional artificial-intelligence challenges of representation and reasoning rear their heads here in all their glory. Quality of heuristics and applicability of evidence, for instance, are subtly—but importantly—different in this reasoning task, and data rarely provides absolute proof or refutation of any particular hypothesis.

Addressing the challenges raised in the previous paragraph required appropriate software-engineering solutions, appropriate artificial-intelligence solutions, and effective integration of the two. From the software-engineering standpoint, it is critical to produce prototypes early on in these kinds of multidisciplinary research settings. Doing so not only brings out any hidden disconnects between the computer scientists and the domain scientists, but also requires that the associated issues be faced squarely. Explicit and implicit disconnects have important implications for the creation of software requirements, as well as for the design and implementation of the software. Perhaps the most important software-engineering related insight that came out of the ACE project is the depth of the need for flexibility in all phases of the software development, and that is one of the central stories of this paper. A key requirements-related problem, for example, is that the computer scientists in such collaborations are neophytes in the other participating scientific disciplines. This lack of knowledge imposes significant start-up costs to the project and increases the need for flexibility in the initial phases of software development. For instance, in the development of ACE, our geoscientist collaborators did not reveal a critical detail concerning calibration and how its results fed into the dating workflows until late in the design process. This new requirement triggered a major redesign in how supporting data sets and input parameters were stored and generated in ACE and how the calibration and dating workflows linked together. All of this is discussed at more length in Section 3.1.

From the AI standpoint, the main challenges in the ACE project were to design an appropriate reasoning engine, populate its knowledge base with information that accurately captured expert reasoning about paleolandforms, and communicate effectively about the results with those experts. Interviews and joint work sessions made it clear that geoscientists proceeded by considering multiple simultaneous hypotheses, examining all available data for even vague hints to support or refute each one. They dealt in terms of the overall weight of evidence, rather than in any certainties. The number of solved problems in cosmogenic isotope dating analysis is relatively small, making a machine-learning or case-based approach impractical. The Bayesian approaches that work so well in so many areas of AI were inappropriate here because they are based on forward probabilistic models for reasoning—not forward *mechanistic* models or backward investigatory models, which are the ones used by geoscientists during forensic reasoning processes. Finally, scientists performing isotope dating are almost always highly trained experts, so a system that presents answers without detailed support is unlikely to be useful. For all of these reasons, we clearly needed a defeasible symbolic system with partial support of some sort. Since geoscience experts communicated naturally—with us and with each other—in an argumentation format, we chose an argumentation-based design for ACE's reasoning engine. The rules that guide this engine's operation are Horn clauses $A \Rightarrow C$ ("A implies C"), annotated with associated confidence and quality judgments about the data or the implication. All of this is discussed at more length in Section 3.2.

Our goal in this paper is to offer the insights that we gained, as a result of the ACE project, into:

- the role of flexibility in software development for multidisciplinary collaborations and
- the representation and reasoning challenges that arise in forensic science, where timelines are unknown, empirical data are scarce, and controlled experiments are impossible

After outlining the general challenges that discovery informatics faces in multidisciplinary research settings, we present a high-level view of ACE: its application domain, its design, its implementation, and its results. We aim to demonstrate how a tight integration of application domain, artificial intelligence, and software engineering expertise is required to address the challenges inherent in forensic paleoclimatology. Without our approach it would have been very difficult to develop an environment that provides true utility to researchers working in this field. The final design environment is now a useful tool that facilitates scientific progress in this important domain.

2. Challenges

Interesting discovery-informatics challenges abound in multidisciplinary research settings. A representative subset that occurred within the context of the ACE project is:

1. Complex Application Domains. Application domains in multidisciplinary research settings are complex. It can require years of experience to master the concepts and techniques needed to work within them.

2. Non-Traditional Specifications. Traditional approaches to requirements-gathering break down when conversation alone cannot lead to actionable software requirements. In such settings, the software engineer must also make use of non-traditional requirements specifications including existing (perhaps poorly written and/or poorly designed) software, as well as research papers drawn from the application domain. Since this requires the software engineer to be up to speed in the application domain—and the application domain is complex—this leads to significant delays in creating software specifications that can be used as the basis for creating a design and implementing a prototype.

3. Technical but not "IT Savvy" Collaborators. Our geoscience collaborators were technical people, highly skilled in mathematics, chemistry, physics and the like, but they were not familiar with the range of options for developing software systems and the implications for choosing one option over another. Computer scientists can develop a blind spot in such situations and be tempted to "dumb down" the design of a software prototype to compensate for the perceived "deficiencies" of their collaborators. Such a move can needlessly limit the range of design solutions considered and ultimately lead to a software prototype that does not realize the full potential of the multidisciplinary collaboration.

4. Incomplete Requirements. The flip side of the previous item is that in multidisciplinary settings, the non-computer scientists can decide that the computer scientists are "not ready" for certain functional requirements and thus withhold those requirements as they "handhold" the computer scientists through the more basic requirements. This situation leads to an incomplete view of the overall set of requirements, which can then lead the computer scientists down the wrong design path.

5. Heterogeneity of Computing Platforms. Significant discrepancies often exist between the computing platforms and computing resources being used by different segments of the multidisciplinary team. Often, the non-computer scientists have access to more-powerful computing resources (e.g., clusters and supercomputers at national laboratories), while the computer scientists primarily employ desktop/laptop devices. This disparity can lead to significant design challenges concerning the software architecture of the software prototypes being developed by the team, along with what programming languages to use, what third-party libraries to acquire, etc.

6. Representations. Expert geoscientists make distinctions between the quality of evidence (e.g. degree of statistical significance) and the quality of heuristics (e.g. extrapolations from accepted theories vs. largely unfounded heuristics). Handling these distinctions is crucial in solving this forensic reasoning problem.

7. Reasoning. Experts in any field are subject to real-world constraints regarding the availability of data. The samples used in the work described here, for instance, are heavy, the landforms involved are often very distant, and the lab work is expensive. However, experts also routinely engage in rich chains of reasoning based on these extremely limited amounts of data. One challenge for this project was to accurately reproduce this reasoning in an automatic way.

8. Knowledge engineering. Implicit assumptions and reasoning are classic hard problems in automated reasoning research, particularly in an interdisciplinary setting like this.

9. User interface. Communication of the results of an automated reasoning system in a way that is effective to human domain experts is another well-known challenge.

This list is by no means exhaustive, but it includes the challenges that were most formative in this project—and those that we feel are likely to be most common in collaborations of this type.

3. Project Overview

The goal of the ACE project, which was developed with NSF ITR funding (Zreda, Bradley, & Anderson, 2003), was to radically advance the state of software tools for the field of cosmogenic nuclide dating. The project's development followed a tight feedback loop between the geoscientists on the team (Zweck and Zreda), the AI specialists (Bradley and de Vesine), and the software engineering group headed by Anderson. The scientific foundation of the dating technique is the accumulation process of certain rare isotopes produced by secondary cosmic rays. The intensity of cosmic radiation is high enough to penetrate the atmosphere and interact with nuclei of atoms in the top few meters of the solid earth, but not strong enough to penetrate to greater depths. This allows scientists to deduce how long a given rock sample was exposed to the sky. When the concentration of a cosmogenic nuclide has been measured and the production rates are known, the exposure age of the sample can be determined.

Cosmogenic nuclide dating techniques are ideal for dating surface features such as meteor impact sites, earthquake ruptures, lava flows, alluvial fans, terraces and landforms associated with the retreat of glaciers (Desilets & Zreda, 2003). But they pose some SE challenges, such as the need for flexiblity in specifying the order and type of computations to be performed and the ability to add new supporting data sets and new types of computations after the initial development of the software prototype was complete (Anderson et al., 2007). The reasoning involved in interpreting the results produced by these techniques is also challenging. Laboratory costs are high and good samples can be hard to come by, so sample sets are small and noisy. Samples may have been disturbed since the formation of the landform, and calibration curves do not represent ground truth. These are some of the challenges that ACE's AI system, Calvin, addresses (Anderson et al., 2010; Rassbach, Anderson, & Bradley, 2010; Rassbach, Anderson, & Bradley, 2011; Rassbach et al., 2007).

3.1 Software Engineering for Cosmogenic Isotope Dating

When we started this project in 2003, the available software tools in this area consisted of Excel spreadsheets and simple, batch-process web applications cobbled together by geoscientists to support their work in understanding the samples they collected in the field. While tools like this can provide a certain amount of utility—mostly to their authors—they come with a host of problems. A variety of assumptions and algorithms were hardwired in those spreadsheets and applications. Those assumptions, which are a fundamental part of the scientific contribution of these analyses, were hard to deduce and difficult (or impossible) to change. Often, the embedded algorithms were tailored to a particular nuclide and expected the specific parameters for that nuclide. To use one of these algorithms on a different nuclide, users were faced with the prospect of copying bits and pieces of the previous spreadsheet into a new spreadsheet and then implementing new code to handle the new nuclide and its associated input parameters. This combination of brittleness, inflexibility, and hard-coding was imposing real limitations on the science. Our goal was to remove these barriers by developing a flexible and extensible design environment to support this complex scientific analysis process.

3.1.1 History of Software Prototypes

The history of the software prototypes developed during this collaboration reflects the team's explicit recognition of the importance of flexibility. This history included three prototypes. The primary specification provided by the geoscience team at the beginning of this process was a handcrafted spreadsheet that performed cosmogenic nuclide dating on one sample at a time. To explore, understand, and supplement the requirements that were implicit in this spreadsheet, the geoscience and computer science teams held a series of face-to-face meetings and e-mail conversations. What emerged was that the geoscientists did not just want a new tool that emulated those spreadsheets. Rather, they were interested in specific improvements in the design and functionality of the software and wanted to be engaged in its ongoing improvement over the course of the project. Even though they had written the software themselves, they were keenly aware of its limitations:

- various algorithms were hardcoded in it
- various supporting datasets were hardwired into it
- it made assumptions about three "constants" that were produced by a separate calibration process
- it had grown so large that it was almost impossible to modify

The software engineering team decided to begin by constructing a prototype called iCronus that mimicked the functionality of the existing spreadsheet in order to (a) become familiar with the domain and (b) have something against which to test the new software. This first prototype—whose name was intended to capture its relationship to Cronus, an existing project in the geoscience community (Phillips, 2012)—was a desktop application, written in Java with a simple Swing-based user interface. Working with the understanding that everything in this prototype would eventually change, traditional object- oriented analysis & design principles were eschewed and primary concepts of the application domain were represented as instances of common collection data structures such as maps, sets and arrays. This enabled the rapid change of these data structures as additional requirements and desired functionality came to light as a result of interactions with the geoscience part of the team. The data structure that represented rock samples, for instance, started out as a simple map of attribute value pairs, but eventually transformed into a more complicated "map of maps of maps" structure that was needed in order to handle experiments that tested different nuclides present in the rock.

As our work on the first prototype progressed, the geoscience team members expressed the desire for functionality that went beyond single-user mode and towards support of small workgroups sharing data via a local-area network. These features included notions of public vs. private experiments and unpublished vs. published sets of samples, data and experimental results. Accompanying these concepts were notions of different types of users with different permission levels regarding the access and sharing of data. As a result, the software engineering team decided to move on from the initial monolithic, desktop-based prototype and migrate to a web-based solution that could be accessed via web clients implemented in HTML, CSS & Javascript. We subsequently implemented our expanded notion of the conceptual framework underlying this project as a web application in the Python- based Django web application framework (`djangoproject.com`). Because of political issues in the geoscience community, this second software prototype was called ACE, which stands for Age Calculation Engine (see Fig. 3).

This ACE prototype consisted of a typical three-tier architecture implemented using the standard model-view-controller design pattern that is naturally encouraged by the design of the Django framework. The underlying conceptual framework (see Fig. 5 later in this paper) was implemented as a set of models, with Django handling the details of synchronizing model information to and from the database. The services that implement the operations that users could perform on these underlying models were implemented via a combination of Django views and client-side Javascript. A

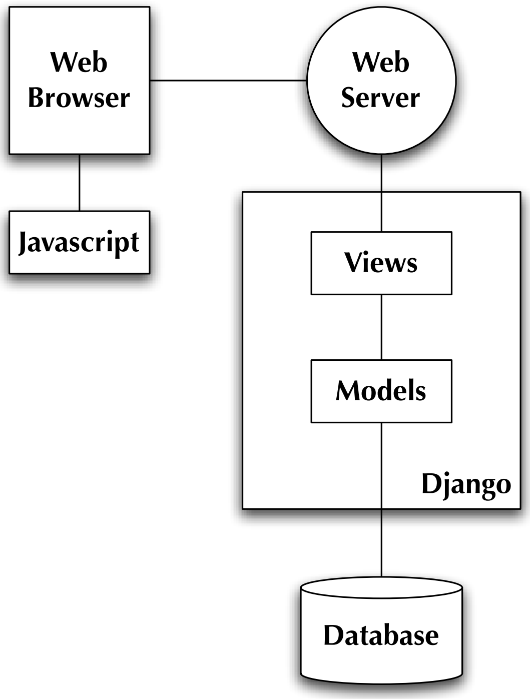


Figure 3. The second prototype was implemented as a web application using the Django web application framework.

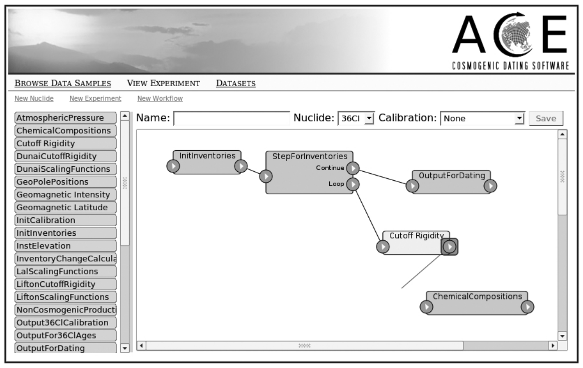


Figure 4. The ACE Workflow Editor of the Django Prototype

Django view is a controller bound to a particular URL within a web application. A view is responsible for generating a response when a browser accesses its URL; typically, the Django template system is invoked by a view to generate HTML that is sent back to the browser for display. When a user edits an ACE workflow, for instance, a workflow view is used to generate the initial page displaying a graphical view of the workflow (see Fig. 4). Client-side Javascript then handles the user's interactions with that page until he or she is ready to submit their changes back to the server. A Django controller then receives those changes, validates them, and updates the appropriate workflow model, which in turn is saved to the database.

The key evolution within the prototype after moving to a web-based solution was that the algorithms embedded in the initial spreadsheet and subsequently implemented as medium-sized Java-based classes in the initial prototype were finally split into small units of reusable python-based software components in the second prototype. A key factor behind this change was that our colleagues were unhappy trying to write new components in the Java programming language. They found Java to be foreign and difficult to understand and even the most technology savvy member of their group encountered major conceptual difficulties when trying to write new components that fit the Java template we had provided. In particular, the need to use a Java class known as BigDecimal was particularly vexing, as its approach to solving problems related to precision were unintuitive and difficult to learn. We knew that the issue was not that our collaborators could not program—they had built the initial spreadsheet **before** our collaboration started after all—it was that their experience with spreadsheet formulas did not prepare them for the statically- typed environment of Java. To provide a sense of the technical skills of our collaborators, consider an example of one of the formulas contained in the initial spreadsheet:

```
=IF (B102<4000,
  VLOOKUP(FLOOR(B102,500),paleomag_table,3) +
   (B102-(FLOOR(B102,500))) *
   (((VLOOKUP(
       CEILING(B102,500),paleomag_table,3)) -
     (VLOOKUP(
       FLOOR(B102,500),paleomag_table,3))) /
    (CEILING(B102,500)-(FLOOR(B102,500)))),
  VLOOKUP(FLOOR(B102,1000),paleomag_table,3) +
   (B102-(FLOOR(B102,1000))) *
   (((VLOOKUP(
       CEILING(B102,1000),paleomag_table,3)) -
     (VLOOKUP(
       FLOOR(B102,1000),paleomag_table,3))) /
    (CEILING(B102,1000)-(FLOOR(B102,1000)))))
```

This line uses conditional constructs, lookup tables, and non-trivial mathematical constructs to determine the age of a rock sample, based on the presence of certain cosmogenic nuclides. In this particular snippet, the code is treating the effects of Earth's magnetic field differently for samples younger than four thousand years old. This spreadsheet consisted of many different pages that contained elementary data sets, complex subcalculations that were accessed via lookup tables from the primary workflow, and circular references that forced the spreadsheet to iteratively process the workflow until the change in a particular value converged to a specified threshold.

As the SE team needed our geoscientist collaborators to be able to generate new types of components that could be added to ACE, we exposed them to a range of alternate programming languages. They responded most positively to the syntax and "feel" of Python. Once they understood that the cell references that they wrote in spreadsheets could be replaced with easy-to- understand variable names, they were off and running inserting working python code into the Python-based component template that we provided. For instance, they soon became comfortable generating code that looked like this:

```
def calculateQ_s(self, s):
  # calculate (LAMBDA_f/Zs) *
  #  (1-EXP(-1*Zs/LAMBDA_f))
  div1 = self.LAMBDA_f / s["Zs"]
  div2 = -s["Zs"] / self.LAMBDA_f
  sub1 = 1.0 - math.exp(div2)
  s["Q_s"] = div1 * sub1
```

Here a rock sample (modeled as a key-value dictionary) is passed to a method that computes the way a sample shields itself from cosmic rays (Q_s) as a result of its thickness (Zs). In ACE, these methods are placed in components that accept a set of samples, process them, and then return them for processing by other components. The associated support datasets are made globally available to all components in the workflow. This structure made it easy for the scientists to extend the functionality of the system without the intervention of a programmer.

With the migration away from the non-component-based spreadsheet and the coarse-grain, difficult-to-understand component-based Java prototype, our team was able to produce fine-grained python components that encapsulated just the functionality that could theoretically be swapped out for a different component that computed a particular data value using a different algorithm. As an example, in the deployed version of ACE, there are different components that calculate the effect of sea level upon the cosmogenic age of a sample based on the different sea level data sets that exist in the literature. When running an experiment on a set of samples, the geoscientist has the option of processing the samples multiple times each using a different sea level calculation and then comparing the results. As this example indicates, the benefit of having these small components is that we could now arrange them into directed graphs and execute them on demand using workflow-based techniques. This was a critical step in addressing many of the shortcomings of the original spreadsheet and allowed much more experimentation and "what if" exploration of the types of algorithms that can be used to date samples using cosmogenic techniques. Our particular implementation of component-based workflows will be discussed in detail below.

After deploying the web-based solution and having our colleagues try it out, they began to voice concerns over where their "data" was stored. Prior to working with us, our geoscientist collaborators always had tight controls on their data, especially unpublished sample data and experimental results. This data had typically been stored on their laptops or desktops with very little chance of that data being accessed by members outside of their research group. With a move to a web-based solution, they realized that they may have to migrate to a situation in which their unpublished data was stored on a server and not on their own personal machines. While we deployed the web-based prototype such that it was accessible only to their group, they still felt nervous about losing control over their data. We'll discuss this issue in more detail below but it serves now as an example of the type of issue that can arise in multidisciplinary settings. This particular issue was considered significant enough that they mandated that we abandon the web-based approach altogether and return to a desktop- based solution, this time implemented in Python but now with the benefit of the workflow technology that had been implemented for the second prototype. In addition, the switch to Python had been validated when our colleagues discovered the "scipy" (a.k.a. scientific python) set of Python modules such as numpy (numerical processing), scipy and matplotlib (a reimplementation of the many plotting capabilities of MATLAB). The resulting prototype described in detail below (and shown in Fig. 2) benefited from the fact that our colleagues could move ahead with their own research confident that any software they created using the scipy modules could easily be migrated into the new software prototype.

The lesson learned with respect to moving to the third prototype was that in a multidisciplinary setting it is critical to find ways that enable the team members to work in parallel. If the team feels that they are stuck waiting on the software engineers to produce usable prototypes, productivity and morale can plummet. Having arrived at the design for the third prototype and having gained valuable feedback and understanding about the domain via the previous two prototypes, we were finally in a position to allow all members of the team to move forward on individual and joint research goals.

3.1.2 Enabling Flexibility and Extensibility in ACE

The key design goals of the third ACE prototype—the one that was ultimately released to the cosmogenic nuclide dating research community—was to create an environment that made no assumptions about the types of nuclides, scaling factors, algorithms, etc. used by a research group and instead allow these elements to be plugged into the environment as needed, and to produce an architecture for the ACE prototype that provide maximum flexibility and extensibility to its users.

The key design characteristics of ACE that enable maximum flexibility and extensibility include the ACE data model, the ACE repository format, the ACE workflow engine, and the set of tools that ACE provides to the geoscientist. When designing each of these elements, we made choices that valued simplicity and extensibility. In addition, we aimed to keep the entry barrier to adopting the system as low as possible. This system was designed from the start to maximize the chance that it would be adopted by the cosmogenic nuclide dating community. We did not want to limit its adoption by artificially imposing the use of a particular database or computing platform. While we did restrict our users to the use of the Python programming language, we did so only after

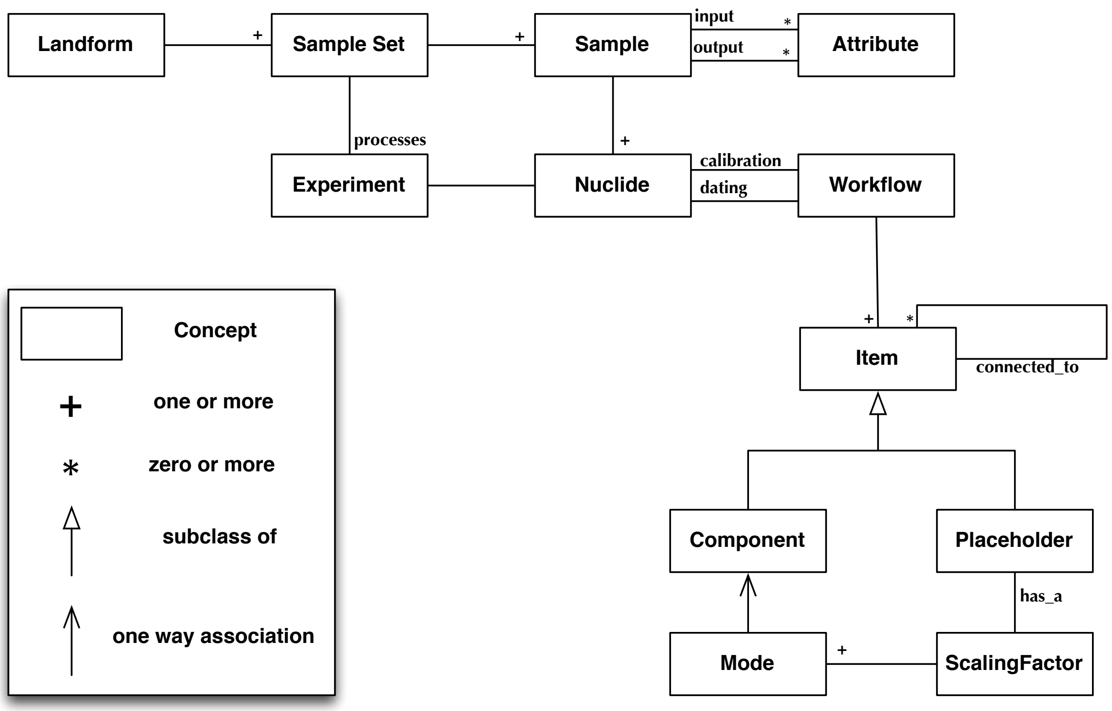


Figure 5. The ACE Conceptual Framework

careful consideration with our colleagues, ensuring they could make the leap from Excel macros to Python-based components and obtaining their assurances that the rest of their research community could likely make that transition as well. Our choice of Python was parsimonious with our other design goals: it is a simple, extensible language that runs on multiple computing platforms, is freely available, and is supported by an active development community.

The ACE Data Model. The underlying data model for ACE is based on a simple conceptual framework that exactly captures the concepts needed to perform cosmogenic nuclide dating (see Fig. 5). Geoscientists use cosmogenic nuclide dating to understand the evolution of a *landform*. Each landform can produce one or more *sample sets*. Each sample set contains one or more *samples* (rocks). A sample can contain one or more *nuclides* and has a set of *input attributes* and a set of *output attributes*. Input attributes are either *nuclide-independent* (present for all samples) or *nuclide- dependent* (present only for samples that contain a particular nuclide). Output attributes are calculated by *components*.

Samples are processed by *workflows* that consist of one or more *components* and/or *component placeholders*. Each component consists of a set of *input ports* and a set of *output ports* and performs a specific computation. Components can be connected together via their ports to form workflows with sequential, branching, and looping paths. A component is activated whenever a sample set arrives on one of its input ports. The component will iterate over each of the samples in the set, perform its calculation, and store the results as one or more output attributes on the sample. Once a component has finished processing a sample, it will place it within a sample set associated with one of its output ports. The component will then pass these sets to the components connected to its output ports, activating them in turn.

A component placeholder is a location in a workflow associated with a particular *scaling factor*. A scaling factor is a calculation that can influence the results of a calibration or dating workflow. For instance, since the earth's sea level has changed over time, a dating workflow that takes sea level changes into consideration will produce different exposure ages than a dating workflow that does not. Q_s above is another example of a scaling factor, since geoscientists might want to consider different ways in which the thickness of a sample might impact its exposure age. Each scaling factor can have multiple *modes* in which a different algorithm or a different data set is used to perform its calculation. A different component is associated with each mode of a scaling factor and that component is plugged into a workflow at the location of its associated component placeholder. Indeed, a workflow cannot be executed until a component has been plugged into each of its component placeholders, which corresponds to selecting a mode for each of the workflow's scaling factors.

An *experiment* has a set of input parameters and an associated nuclide. Each nuclide has an associated *calibration workflow* and *dating workflow*. These two workflows must both have the same set of scaling factors. Indeed the only difference between these workflows is that a calibration workflow is used to calculate a set of *production rates* that are used by the dating workflow to calculate exposure ages of samples. After an experiment's nuclide has been selected, a user must select a mode for each of the scaling factors that appears in the nuclide's two workflows. At that point, the calibration workflow can be applied to a *calibration data set* to produce the production rates needed by the dating workflow. A calibration data set is simply a sample set whose samples have been dated using another dating technique (such as carbon-14 dating). Once the production rates have been calculated, the experiment's dating workflow can then be used to process any sample set imported into ACE.

It should be clear that this data model provides a wide range of customization points for geoscientists performing research on cosmogenic nuclide dating techniques. They can create variants on existing workflows for existing nuclides by creating a new mode for a particular scaling factor. They can create new workflows for previously unsupported nuclides. They can incorporate new calibration sets into the environment. They can encapsulate a new type of calculation within a new component and then include that component into existing or new workflows. Granted, there is some complexity here but that is a reflection of the problem domain itself. Within these carefully chosen concepts, however, a great deal of flexibility and extensibility is possible and this has contributed to the fact that ACE is still in active use by this research community, years after its original developers ended development on ACE.

The ACE Repository Format. ACE makes use of a simple, extensible repository format that consists of a hierarchical set of folders and text files. This format was chosen because it was human readable, easy to archive, and easy to modify. Scientists nervous about losing data as a result of trying out a new workflow can simply save their repository, quit ACE, and then create copies of the repository by duplicating the directory in the filesystem for safekeeping. In addition, an input error made while using ACE's graphical user interface can easily be fixed by finding the appropriate file in the repository and editing it with a text editor outside of ACE. Our colleagues were very enthusiastic about these capabilities and immediately were able to add new supporting datasets and samples to their repositories in support of their research.

For instance, supporting datasets might include standard scientific constants, sea level data, sunspot data, metrics related to the periodic table of the elements, and geomagnetic intensity data. All of these sets were stored inside of text documents following a standard format in which the name of the collection appears on the first line of the file followed by the data itself on the second line of the file stored as a collection (typically a map or array) in a Python-readable format. ACE provided tools in its graphical user interface to define the structure of these collections and then to import them into the environment automatically from CSV files. However, advanced ACE users can generate the required files directly and simply place them in the appropriate spots in the repository to access them from within workflows and other ACE tools.

Similar techniques were used to store information on components, workflows, scaling factors, modes, experiments, nuclides, and samples, as well as other data related to how this information was presented by ACE's graphical user interface. The flexibility of this repository format contributes to the ability of ACE's community to continue to make use of ACE as new techniques are developed and new types of supporting data sets are used to improve the overall accuracy of the dates produced by ACE.

The ACE Workflow Engine. The only way in ACE to calibrate nuclide production rates and calculate the ages of newly collected samples is via the use of ACE's workflow engine. A workflow is always executed within the context of a particular experiment. The experiment points at a particular workflow description and a set of specific modes for the scaling factors (such as the influence of sea level or sunspots on the production of a particular nuclide) that appear in that workflow. The workflow engine reads in the workflow's description, a simple text file that specifies the connections between a set of placeholders or components written in Python and wires up the specified components dynamically at runtime. If the engine encounters a placeholder for a scaling factor—essentially a token in the workflow description that means "insert sea level calculation here" —it consults the experiment for the selected mode of that factor and inserts the appropriate component, or set of components, into the appropriate spot. At that point, the workflow reads the experiment's selected sample set into memory and passes it to the first component of the workflow, executing the workflow until all samples in the sample set have been processed.

The flexiblity of this approach is extremely powerful. No calculation is hardwired into a dating or calibration workflow. Components can be easily swapped in and out depending on the needs of the experiment. New workflows can be created by starting from scratch and specifying an all new set of connections between existing and newly written components or an existing description can be copied, renamed, and modified to suit a researcher's needs. These edits can be performed in a text editor by accessing the files contained in the workflows subdirectory of ACE's repository format.

In the future, the computational model of the workflow engine can be transformed from a single threaded model to one which performs its calculations in parallel. Each component represents an atomic computation and that computation can be applied to all of the members of a sample set in parallel. In this model, workflows become a sequence of "map jobs" (after the terminology of MapReduce) that need to be invoked on the right data (samples) at the right time. In the third prototype, we chose to deploy the single threaded model of computation for two reasons; the first is in keeping with our design principle of simplicity—a single threaded model of computation is easier to understand and debug—while the second is that the sample sets of this research community tend to be very small numbering in the tens of samples per set and thus the need for parallel computation—and its increased complexity—is greatly reduced. Most workflows complete in a matter of minutes and these execution times are considered acceptable by this research community.

The ACE Tool Set. The set of tools provided by ACE to its users include a sample browser, an experiment editor, and various support utilities to import new data sets and plot the results of workflows in various ways. The set of tools that can be integrated into ACE is unbounded. This extensibility is enabled by ACE's data model and simple repository format. ACE provides a set of APIs that can read the contents of a repository into memory as a set of instances of the classes contained in ACE's data model. A new tool for ACE's graphical environment has to follow a set of conventions for inserting itself into ACE's menu system and be able to interact with ACE's data model via ACE's API. This approach was used to create the core sample browser and experiment editor by ACE's primary developers and has since been used by ACE's research community to add new tools that generate new types of charts and reports or that perform additional analysis on samples after they have been dated by one of ACE's workflows.

As a result of these careful design choices, we have produced a flexible, extensible design environment for new cosmogenic nuclide dating techniques. The set of tools and techniques described so far are very powerful and would indeed serve the basic needs of ACE's research community for many years. However, ACE has been further enhanced with the integration of Calvin, ACE's reasoning engine, which aims to help ACE's users make sense of the ages calculated by ACE. It is these sensemaking features that transforms a capable system into an exceptional tool for geoscientists in this research community.

3.1.3 Evaluation

At the beginning of this project, our collaborators provided us with an Excel spreadsheet that implemented a single cosmogenic dating technique for samples containing the nuclide ^{36}Cl. The spreadsheet could only process one sample at a time and had its production rates hardwired into it. That is, the spreadsheet did not contain any functionality to handle the process of calibration. Furthermore, the dating algorithm implemented by the spreadsheet had a certain set of scaling factors built into it with no flexibility to change their algorithms or to add or remove new scaling factors.

Our design environment can now be used to design an experiment that exactly mimics the functionality of the original spreadsheet. That configuration, however, is just one of an almost unlimited set of experiments that can be specified, executed, and analyzed by the current design environment. The flexibility built into our concept of workflow allows any number of calibration and/or dating algorithms to be constructed, each of which can support multiple combinations of scaling factors and other parameters and variables. A particular scaling factor can easily be adjusted by adding a new "mode" to it and supplying a component that performs the calculations associated with that mode. Our design environment smoothly handles the process of calibration and ensures that dating workflows use production rates calculated by calibration workflows that have the same structure as the dating workflows and the same set of scaling factors. In addition, the design environment can store multiple sets of samples and can apply a dating workflow to multiple samples at once. Furthermore, our design environment provides plotting and analysis services over the samples that it has processed, allowing our users to perform productive work within the environment. Finally, it allows users to export its data for postprocessing or dissemination.

The "look and feel" of the ACE design environment has been greatly influenced by the feedback of our team of geoscientists. ACE was incrementally deployed to our collabortors during development so they could use it on a daily basis, even when the software had only a few features implemented. As new features were finished, or the user interface was enhanced or changed, new versions were deployed so the geoscientists could try out the latest version of the design environment. They would then provide feedback which we would attempt to incorporate in subsequent releases. This incremental, iterative process is exactly the type of development life cycle that is recommended when attempting to create a design environment for a community of users (Fischer et al., 1994).

As a result of these design, implementation, and evaluation choices, our cosmogenic dating environment demonstrates how modern software engineering principles and techniques can be used to produce flexible and extensible software that meets the needs of its users. Our initial evaluation has compared the capabilities of the ACE design environment with the functionality of the software it was built to replace (as well as with other cosmogenic dating software). It represents a proof-of-concept that the limitations of these previous systems were artificial and that modern software engineering techniques can be used to produce a system that provides the flexibility needed by geoscientists to perform cosmogenic dating research. No other cosmogenic dating software offers the same functionality, flexibility and future extensibility as is provided by ACE. Indeed, this statement is supported by the statistics of use presented in this paper's abstract. The ACE systems has been downloaded nearly 600 times and has remained in active use years after we stopped active development on the software.

3.2 Automated Reasoning about Cosmogenic Isotope Dating

The complicated series of calculations involved in dating individual samples that is described in the previous section is only the first step in cosmogenic isotope dating of paleolandforms. The next step is to reason from those results—that is, the exposure times of the samples—in order to understand the overall history of the landform. This is one of the most challenging and important problems that cosmogenic isotope dating specialists regularly solve, generally requiring weeks or months of effort on the part of a human expert (and sometimes another trip to the field to gather more samples). If all of the sample exposure ages overlap, the problem is comparatively easy: the true age is somewhere in that overlap. This rarely happens, however; rather, the spread of the exposure ages is generally broad and uneven. In these cases, the scientist must construct a geologically meaningful and defensible explanation for the spread in order to deduce the true age of the landform.

ACE's reasoning engine, Calvin, automates this complicated, subtle reasoning process (Anderson et al., 2010; Rassbach, 2009; Rassbach, Anderson, & Bradley, 2011; Rassbach & Bradley, 2008; Rassbach et al., 2007; Zweck et al., 2012). This engine is an iterative argumentation system that is based primarily on the Logic of Argumentation of Krause *et al.* (Krause et al., 1995). It was hand-written for this application, can handle both discrete symbols and continuous values, and consists of roughly 500 lines of Python code. Its knowledge base incorporates more than 100 rules, gleaned from an extended knowledge-engineering process involving dozens of geoscientists. Its input is a set of samples that have been dated by the machinery described in the previous section. Its goal is to abduce what process(es), acting over what time periods, could have produced that set of sample ages. Calvin explores this forensic scenario space by enumerating all possible hypotheses about the processes that may have affected the landform, then considers all the evidence for and against each one. Testing of each individual hypothesis involves generating all possible arguments for and against it. To do this, Calvin first finds all of the rules in its knowledge base that apply to that hypothesis, then unifies those rules with the sample ages and uses that unification to construct a collection of arguments about the associated conclusion (viz., moraine X has been affected by processes Y and Z).

Calvin's design was guided by the nature of the problem at hand, which involves heuristic reasoning with partial support, frequent contradictions, and sparse, noisy data. Most of the explanations that experts find for the apparent age spread of a set of samples come from a short, known list of geologic processes, most commonly exhumation/erosion and inheritance—when a landform 'inherits' one or more older rocks. It is also important to consider the possibility that no process was at work. Despite the relatively small number of candidate processes, constructing these explanations is not a simple matter. Available data are noisy and may not be trustworthy. Different processes may have similar (or cancelling) effects, and multiple processes may be at work. The reasoning involved is heuristic and conclusions do not have absolute confidence; stronger arguments against them may be found, and the current best hypothesis overturned. (This is the main distinction between argumentation and classical first-order logic systems or "expert" sytems.) Moreover, these heuristics are often vague or only slightly supportive of their conclusions. For example, several experts informed us that they "prefer the explanation that requires throwing out the least data." This heuristic expresses a preference, not a certainty, but it obviously lends some weight to the discussion. Implementing partial support of this nature has been a traditionally slippery problem for AI (Stenning & van Lambalgen, 2008). Reasoning about sample ages generally involves a fair amount of evidence both for *and against* several processes. Handling this type of contradiction is another well-known problem for automated reasoning (Stenning & van Lambalgen, 2008).

Contradiction is an extremely important issue in this application. The heuristics used in reasoning about cosmogenic isotope dates frequently contradict each other, and different experts also hold contradictory opinions about the correct heuristics. Especially interesting cases of contradiction arose when experts contradicted themselves. During the knowledge-engineering phase of the project, one geologist said

> "The thing about inheritance is, it's usually thought about as quantized, not incremental. So in Antarctica, say ice advanced every 100k, so a sample is 20k or 120k, not 21k. So that is one thing that should be commonly true about inheritance, it should reflect events, should be things that date from past advances, should be quantized."

And in the next breath, he said

> "However, you can convince me you would see a continuum, [...] the glacier advanced and quarried exposed material to different depths, so delivering stuff with 100k age but could be at depth and look like only 70k. So if you have stuff with one past event but chop it up and deliver different parts you might not see the quantized aspect."

That is, not only do experts disagree with each other, they sometimes disagree *with themselves.*

Finally, effective interaction with users was also an important consideration in Calvin's design. Conversations between the AI and geoscience members of the team made it clear that the latter would find most useful a system that not only presented its reasoning in full, but also reasoned as they did, using the same knowledge and information. In addition, we learned in later interactions that educating new students in this reasoning process is a major challenge for geoscience professors, futher indicating the usefulness of a system that provides a complete chain of its reasoning.

Calvin's answers to these challenges begin with the design of its engine. Geoscientists, we have found, find it natural to reason via arguments: *for* results with which they agree, and *against* those with which they disagree. (One landform dating expert even told us "Well, mostly what we do is argue with each other.") This maps naturally onto the venerable method of multiple simultaneous hypotheses (Chamberlain, 1965): scientists construct a list of possible scenarios, then attempt to form complete arguments for and against every hypothesis, and find themselves convinced by the argument with the strongest support. Argumentation, which has a long—albeit mostly theoretical—history in AI, was an obvious choice for ACE's reasoning engine because of its natural match to this process. Other traditional AI strategies cannot handle some of the unique ways in which geoscientists reason about cosmogenic isotope data. Experts work with multiple contradictory heuristics at the same time, for instance, and several weak arguments can weaken and/or defeat a strong one, which requires some novel modifications to traditional argumentation strategies (and completely rules out traditional knowledge-based or "expert" systems).

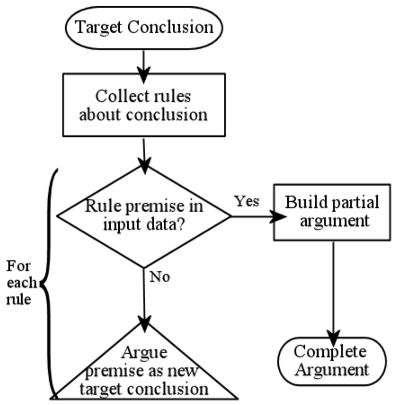

Target Conclusion
Collect rules about conclusion
Rule premise in input data?
Yes
Build partial argument
No
For each rule
Argue premise as new target conclusion
Complete Argument

Figure 6. A flowchart of Calvin's backwards-chaining rule-unification process. For a target conclusion, Calvin finds all the rules that might apply to the conclusion. Then, for each rule, if the rule premise can be shown (or falsified) from the input data, Calvin uses those data and the rule to build a partial argument about the conclusion with a confidence value. Otherwise, Calvin sets the rule premise as a new target conclusion and repeats. Once all rules have been unified, Calvin combines them to form a complete argument about the original target conclusion.

Calvin's ruleset was the keystone of the AI effort in this project. The design of these rules was critical; noise and partial support, for instance, were addressed by allowing support for hypotheses to have variable strength. Another critical insight in Calvin's rule-design strategy is that not only can specific *knowledge* be more or less certain, but the *evidence used to apply the knowledge in a specific case* may be of variable suitability. For example, the statistical significance of a sample property might cross the 0.1σ, 0.05σ, or 0.01σ boundary; each of these indicates a stronger case for an argument based on that property holding true. For this reason, Calvin uses a rich, multi-level representation to capture experts' confidence in the data and in the conclusions drawn from it, and to propagate that knowledge through the forensic reasoning chain. These design elements are discussed in turn in the remainder of this section; the knowledge engineering process through which these rules were crafted is described in the following section. More detail on all of this material can be found in (Rassbach, 2009; Rassbach, Anderson, & Bradley, 2011).

Using a list of the processes that are known to operate upon landforms—erosion, snow cover, and so on—Calvin begins by generating a set of candidate hypotheses about what affected the input samples. It then considers these hypotheses one at a time, building arguments for and against each one using backwards chaining. The first step in constructing an argument involves finding all the rules that apply to that hypothesis—i.e., those that refer to the same conclusion. The engine then applies unification to each of these rules, which either produces a new conclusion to consider or generates a comparison to input data. A flowchart of Calvin's backwards chaining process in Figure 6. Figure 7 shows an example of the logic flow of Figure 6 as applied to the scenario on page 19: the discussion with the expert about inheritance producing either "quantized" ages or a smooth drop-off in sample ages. Beside each rule is a quality rating recording the expert's expressed confidence in that piece of knowledge: the quality of the rule in which the expert expressed more confidence is higher. Calvin considers each of these rules in turn (and any other rules in its knowledge base about

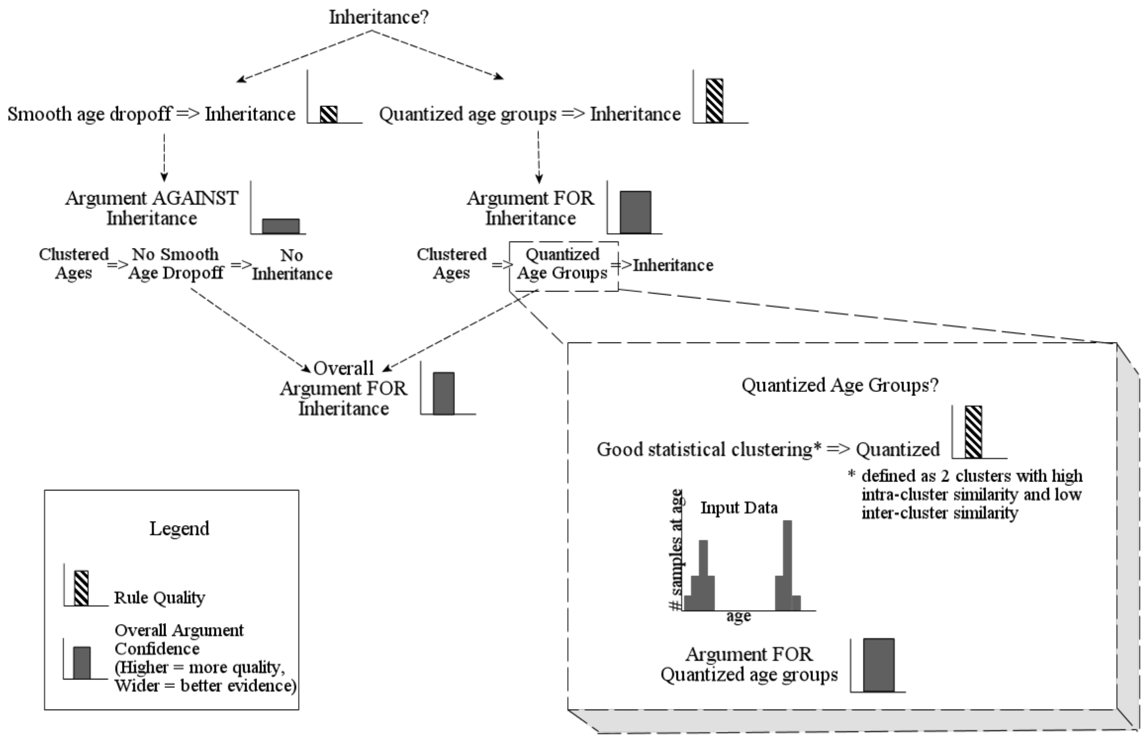

Inheritance?
Smooth age dropoff => Inheritance
Quantized age groups => Inheritance
Argument AGAINST Inheritance
Argument FOR Inheritance
Clustered Ages => No Smooth Age Dropoff => No Inheritance
Clustered Ages => Quantized Age Groups => Inheritance
Overall Argument FOR Inheritance
Quantized Age Groups?
Good statistical clustering* => Quantized
* defined as 2 clusters with high intra-cluster similarity and low inter-cluster similarity
Input Data
# samples at age
age
Argument FOR Quantized age groups
Legend
Rule Quality
Overall Argument Confidence (Higher = more quality, Wider = better evidence)

Figure 7. An example of Calvin reasoning about inheritance (using two rules about inheritance). Each rule is shown with a quality rating; better knowledge has a higher rating. Arguments are shown with a complete two-dimensional confidence, taking into account both rule and evidence quality. The inset box shows a sub-argument used to extract the conclusion that the input data fall into quantized age groups, which is then included as part of the argument about inheritance. The two conflicting arguments are combined to form one overall argument about inheritance, with a confidence level in the final conclusion that reflects the confidence in both the proponent and detractors.

inheritance). If a rule's premises cannot be found in the input data—e.g., that ages are quantized using a statistical measure, in the example of Figure 7—Calvin then argues about those premises as new conclusions. As these rules are applied, the engine assigns the resulting arguments a confidence rating based on the quality of the rule and the applicability of the evidence to the rule (e.g. statistical significance). Finally, Calvin builds an overall argument—in this case in favor of inheritance—from the arguments generated using each rule. The final argument includes any detractors found during the process, so the user can see they have been taken into account. The engine reports an overall confidence in whether inheritance has occurred based on all the contributing arguments (more on this below).

Every rule in Calvin contains both a conclusion and a template for evidence that supports that conclusion. The primary portion of a rule is an implication of the form $A \Rightarrow C$, where A may be either a single literal or the conjunction (or disjunction) of several literals, and C is the conclusion that A supports. The two contradictory statements on page 19, for example, became two different rules in Calvin's knowledge base: one that looks for a smooth increase in sample ages as evidence for inheritance, and one that looks for "quantized" inheritance (defined as highly clustered ages). These rules directly contradict each other; in fact, the example in Figure 7 shows that the exact same input data lead to both an argument for and an argument against inheritance. Calvin's confidence system allows it to sort out this contradiction and thereby reproduce this expert reasoning accurately—i.e., to disagree with itself.

Given a rule, Calvin forms an argument—not a proof—for each element in A, then uses those arguments to create an overall argument for C. An argument is thus composed of expert knowledge (the rule) and the portion of the input data to which that knowledge applies. The representation of an argument contains both the rule and the arguments for the antecedents. Calvin's rules also contain a quality rating, expressing expert confidence in the rule as recorded. The quality rating of a rule and the strength of the applicable evidence (e.g. statistical significance) are used to judge the relative and absolute strengths of arguments. Calvin's backwards-chaining engine generally makes no distinction between negative and positive evidence. This is not a valid method in classical logic, where the knowledge that $A \Rightarrow C$ certainly does not imply that $not(A) \Rightarrow not(C)$. However, Calvin's reasoning is intended to mimic that of human experts, who not only apply rules in this negative fashion[1], but even regard it as a sufficiently defensible practice that they discuss it in published reasoning. For example, Jackson *et al.* (Jackson et al., 1997) state that, since there is no visual evidence of erosion, erosion is unlikely in the area under consideration.

Cosmogenic isotope dating experts have firm ideas about confidence in different measurements, processes, and conclusions, and those ideas play important roles in the analysis process. Calvin's assignment of quality ratings directly to rules allows it to capture and operationalize this information. When building an argument using a rule, Calvin combines the rule quality with the applicability of the knowledge used to build a two-dimensional confidence vector. The insight behind this solution is that not only can specific knowledge be well-regarded or more tenuous (for example, in the interview snippet above it is clear that the expert regards the knowledge that inheritance should have multiple peaks as higher-status than a notion that it might instead produce a smooth dropoff).

1. Consider the following everyday example: $A =$"natural sunlight" and $C =$"daytime." Here, both $A \Rightarrow C$ and $not(A) \Rightarrow not(C)$ make sense.

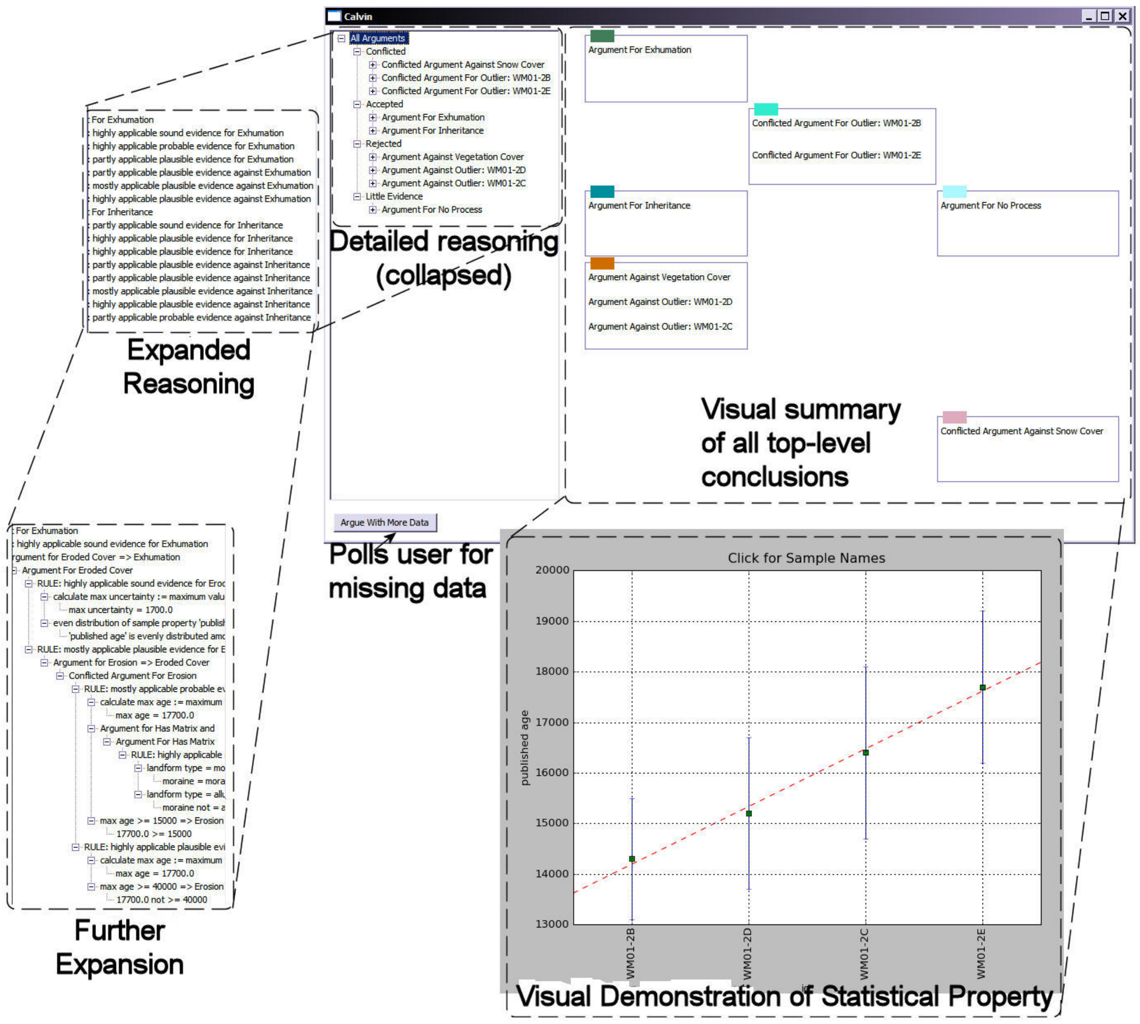
The evidence used to apply that knowledge to the conclusion—the distribution of sample ages, in the example—may also be of variable applicability (consider a distribution of only three or four samples, for example, which should clearly carry lower weight). For this reason, Calvin uses a two-dimensional vector with qualitative (not continuous) values to capture both of these flavors of confidence.

This rich representation of confidence—which is critical to weighing one argument against another—introduces several new research issues: how to set the thresholds and weights, how to weigh the two elements against one another (e.g., a conclusion derived from high-quality evidence and low-quality knowledge versus one from medium-quality values of both), and how those levels should be transmuted as the engine maps them through the rules. Thresholds and weights in Calvin's rules were assigned based on data collected in expert interviews, as discussed in Section 3.2.1. Assigning a confidence to an overall argument based on the confidence in its parts is particularly problematic when arguments of different strength can be made both for and against a given conclusion. Figure 7 shows a visual representation of how these confidences are combined; see (Rassbach, 2009; Rassbach, Anderson, & Bradley, 2011) for a full technical discussion. Calvin's final arguments are shown to the user in a way that incorporates all the information used to make the arguments and determine its confidence level; see Figure 8 for an example screenshot.

Once Calvin has constructed complete arguments for every hypothesis about processes and conditions that might have affected the landform, and assigned a confidence value to each of the considered conclusions, it displays those results to the user in the form of a tree. The conclusion is at the top of the tree; supporting rules are displayed as children. Specific evidence from the inputs and information about Calvin's confidence accompanies each rule. Calvin also displays its overall confidence in each top-level conclusion, as well as in cases of controversy (viz., strong evidence both for and against).

3.2.1 Development & Evaluation

Calvin was developed and evaluated iteratively. Members of the AI team spent a total of roughly 30 days onsite with the geoscience team members over the course of the first four years of the project. These interactions guided the development of Calvin's architecture. This process was challenging and required several iterations to converge; as discussed in Section 2, knowledge engineering is famously hard for many reasons, including implicit assumptions and reasoning—particularly in an interdisciplinary setting like this. The biggest hurdle to understanding the isotope dating field was the desire of experts on the isotope team to present outsiders with an idealized, and arguably more interesting, version of how the reasoning that Calvin was eventually designed to do is done by field experts. Early discussion was limited to (what we later discovered were) extremely unusual projects dealing with large numbers of samples and uncommon processes, without explicit discussion that these were unusual cases. This led to an initial AI design aimed at statistical analysis—something that the average dating project has far too few samples to do reliably—and spatial reasoning. It was only several months later, when this design had been completed and the AI team was presenting it to the geoscience team, that the AI team learned that their understanding of the field was fundamentally flawed. Fortunately, the iteratively planned nature of the project meant that this was an expected feature of the development process, not a catastrophic setback.


Calvin
Detailed reasoning (collapsed)
Expanded Reasoning
Further Expansion
Visual summary of all top-level conclusions
Argument For Exhumation
Conflicted Argument For Outlier: WM01-2B
Conflicted Argument For Outlier: WM01-2E
Argument For Inheritance
Argument For No Process
Argument Against Vegetation Cover
Argument Against Outlier: WM01-2D
Argument Against Outlier: WM01-2C
Conflicted Argument Against Snow Cover
Argue With More Data
Polls user for missing data
Click for Sample Names
published age
20000
19000
18000
17000
16000
15000
14000
13000
WM01-2B
WM01-2D
WM01-2C
WM01-2E
Visual Demonstration of Statistical Property

```
⊟ All Arguments
   ⊟ Conflicted
      ⊞ Conflicted Argument Against Snow Cover
      ⊞ Conflicted Argument For Outlier: WM01-2B
      ⊞ Conflicted Argument For Outlier: WM01-2E
   ⊟ Accepted
      ⊞ Argument For Exhumation
      ⊞ Argument For Inheritance
   ⊟ Rejected
      ⊞ Argument Against Vegetation Cover
      ⊞ Argument Against Outlier: WM01-2D
      ⊞ Argument Against Outlier: WM01-2C
   ⊟ Little Evidence
      ⊞ Argument For No Process
```

```
: For Exhumation
: highly applicable sound evidence for Exhumation
: highly applicable probable evidence for Exhumation
: partly applicable plausible evidence for Exhumation
: partly applicable plausible evidence against Exhumation
: mostly applicable plausible evidence against Exhumation
: highly applicable plausible evidence against Exhumation
: For Inheritance
: partly applicable sound evidence for Inheritance
: highly applicable plausible evidence for Inheritance
: highly applicable plausible evidence for Inheritance
: partly applicable plausible evidence against Inheritance
: partly applicable plausible evidence against Inheritance
: mostly applicable plausible evidence against Inheritance
: highly applicable plausible evidence against Inheritance
: partly applicable probable evidence against Inheritance
```

```
: For Exhumation
: highly applicable sound evidence for Exhumation
rgument for Eroded Cover => Exhumation
⊟ Argument For Eroded Cover
   ⊟ RULE: highly applicable sound evidence for Eroc
      ⊟ calculate max uncertainty := maximum valu
         └ max uncertainty = 1700.0
      ⊟ even distribution of sample property 'publish
         └ 'published age' is evenly distributed amc
   ⊟ RULE: mostly applicable plausible evidence for E
      ⊟ Argument for Erosion => Eroded Cover
         ⊟ Conflicted Argument For Erosion
            ⊟ RULE: mostly applicable probable ev
               ⊟ calculate max age := maximum
                  └ max age = 17700.0
               ⊟ Argument for Has Matrix and
                  ⊟ Argument For Has Matrix
                     ⊟ RULE: highly applicable
                        ⊟ landform type = mo
                           └ moraine = mora
                        ⊟ landform type = allu
                           └ moraine not = a
               ⊟ max age >= 15000 => Erosion
                  └ 17700.0 >= 15000
            ⊟ RULE: highly applicable plausible evi
               ⊟ calculate max age := maximum
                  └ max age = 17700.0
               ⊟ max age >= 40000 => Erosion
                  └ 17700.0 not >= 40000
```

Figure 8. A collection of screenshots from Calvin's interface [after (Rassbach, 2009)]. Users can drill down to any level of reasoning, allowing them to explore and understand the system's arguments around any given conclusion.

In contrast to the serious difficulties involved in learning enough about the field to design a useful reasoning system, it was almost always a trivial exercise for the AI team to turn a piece of expert knowledge into a system rule once that system was designed. Most of the kinds of statements made by experts, such as "when you are working with a moraine your first thought is erosion or inheritance," translate naturally and immediately into Calvin's rule structure, which made the encoding of knowledge a comparitavely easy task. In addition, experts in geoscience (not just on the ACE team) were extraordinarily generous with their time. Some experts were willing to devote the majority of two entire working days to simply discussing their work at a detailed level, and virtually every expert we contacted for an interview was willing to devote an hour or two in the middle of a busy conference to answering our questions. Fortunately, even though all of these experts approached the language and practice of field work in different ways, their reasoning about samples after the lab work was done was remarkably uniform, so a second redesign was not needed.

Once the basic architecture and knowledge representation structures were in place, the lead AI team member (de Vesine) then conducted two onsite demo/interviews—each spanning almost two days—with other geoscientists and then iterated on the design. These experts were recommended by the lead ACE geoscientist for this purpose, based on his knowledge of the field. In these two long-form interviews, these experts made many statements that helped to confirm the validity of Calvin's basic design: i.e., that the results were similar in both structure and content to the reasoning of domain experts. These interactions also fleshed out many of Calvin's existing reasoning chains: e.g., what percentage of samples must agree to result in a high-confidence argument for 'no process'. They also led to a few surprising discoveries: for instance, that visual observations in the field lend significantly more confidence to conclusions than the AI team had expected. Finally, we learned from these interviews that teaching students to perform reasoning about landforms is one of the hardest challenges for professors in this field. Based on this, the AI team altered Calvin's interface to attempt to make results more obvious to students, who could then use it as a learning tool.

Finally, de Vesine attended the main professional meeting of the geoscience research community—the fall 2008 meeting of the American Geophysical Union (AGU)—and conducted 11 one-hour demo/interviews with a range of people working actively in isotope dating[2]. Some spoke English as a first language, some did not; some were veterans of the field and others were graduate students or new postdocs. At this point, the basic design of the representation and reasoning structures was complete; the purposes of these interviews were both to broaden and deepen Calvin's knowledge base and also to assess its overall effectiveness. In these interviews, too, experts made several general statements that emphasized the appropriateness of Calvin's basic design, such as the following exchange:

> **de Vesine:** So the paper has a verified explanation?
>
> **Geologist:** we found some evidence that supported this [...] You look for evidence that supports or falsifies individual hypotheses.

And later in the same interview:

2. Specifically, those whose AGU topics and recent publications in the field's journals concerned cosmogenic isotope dating.

> **de Vesine:** I wanted to ask about how you teach this analysis to new grad students and what they get wrong while learning
>
> **Geologist:** [long pause], that's not an easy question actually. [...] it's very ad hoc [...] they don't understand [some data] at all so you sit them down and talk about them and give them things they could do to test hypotheses: graphs to make and data to collect and we work through it until we are satisfied we have come up with the most reasonable hypothesis

A persistent theme in all of these interviews was how pro-actively experts acknowledged and emphasized the level of disagreement in the field. Not only were they anxious to point out likely rebuttals that other experts would make to their theories, but they also introduced, without prompting, scenarios where they would need sufficient evidence to overrule a colleague's conclusions about a landform. All of the data obtained in these interviews was incorporated into Calvin's knowledge base, which currently includes 108 rules that represent approximately 50 hours (almost 100 transcribed pages) of direct, intensive interviews with more than two dozen experts in cosmogenic isotope dating. These transcripts can be found in their entirety in (Rassbach, 2009).

Our final assessment of Calvin involved using it to reproduce published work: that is, feeding it the data in a published paper and comparing its results against the claims made in that paper. Experts publish a portion of their qualitative reasoning about a landform when they publish a new dataset. While this presentation is usually incomplete due to space limitations and the desire to maintain reader interest, it typically includes information about both rejected and accepted conclusions. This matter is useful for determining if Calvin's recall is sufficiently high for the most important arguments. For this comparison, we found twenty-five randomly selected papers that appeared from their titles to deal with cosmogenic isotope dating. Of these, eighteen actually discussed one or more isotope dating problems in any detail. These publications included a large cross-section of authors and different isotopes, and spanned about ten years, providing a broad basis of comparison. For each of these papers, we extracted every statement that made an assertion, such as the following example from (Jackson et al., 1997):

> Erratic A (sample AE95110101) yielded an age... almost four times older than the next oldest age. This age is clearly anomalous... [t]he most likely explanation for this anomalous age is exposure to cosmic radiation prior to glacial transportation.

We then converted these statements into a form that more closely matched Calvin's terminology. This involved identifying the conclusion argued for in the statement, estimating a level of confidence from terms such as 'clearly' and 'possibly,' and extracting the evidence used in the statement to support the conclusion. Then, we converted the terms in the evidence and conclusion into Calvin's terminology: for example, 'inheritance' instead of 'prior exposure to cosmic radiation' (which is the definition of inheritance). We cross-validated these results by asking two other people (not experts in isotope dating) to perform the same conversions. We then entered all of the data in the paper, ran Calvin, and compared its output to the converted arguments. It closely reproduced the authors' arguments 62.7% of the time and produced similar arguments a further 26.1% of the time. In many cases, the similarity was striking, especially when the authors of the paper expressed significant doubt about their conclusions. See (Rassbach, 2009) for detailed results.

Perhaps the most interesting cases were when Calvin produced an argument that did not appear in the original paper. When examining (Ballantyne, Stone, & Fifield, 1998), for instance, Calvin argued that exhumation was at work. The main evidence for this was a disagreement with ages determined for this landform via other methods. To judge these results, we asked a domain expert to assess Calvin's new argument. He responded:

> I think I see both sides here. From the results, the fact that the ages are younger than the C14 data means that exhumation should be taken very seriously (...) there is not much in the way of material that could bury them. However the peaks themselves are eroding...

In this expert's opinion, then, the lack of explicit discussion of exhumation in (Ballantyne, Stone, & Fifield, 1998) was a major oversight. Although Calvin does not give exactly the same argument, it found a major gap in the reasoning published by these authors.

In all, Calvin provides several contributions to AI and to the larger scientific community:

- Its rule base is an explicit representation of the knowledge of two dozen experts in landform dating
- It incorporates a rich system of confidence that captures the reasoning of real scientists in a useful way
- It is a fully implemented and deployed system—a surprisingly rare thing in the argumentation literature
- It is a real tool that is in daily use by real scientists

Overall, our assessment of Calvin indicates that its knowledge base is largely complete and that missing data is easy to add. Moreover, Calvin's reasoning process is very similar, in both structure and content, to the reasoning process of domain experts. While its design is similar to (and inspired by) previous work, it is unique—and more suitable than those frameworks to implementing the multiple-hypothesis reasoning that is employed by experts in cosmogenic isotope dating. Calvin successfully solves this challenging problem in an intuitive and natural way, following the structure of methodology already in use by domain experts.

4. Related Work

Two software programs are available to date landforms using cosmogenic nuclides. The CRONUS-Earth series of calculators[3] compute sample ages of ^{10}Be, ^{3}He, ^{36}Cl, and ^{26}Al using an online web server operating as a front end for a set of MATLAB routines. An important issue with these tools is that geoscientists must be willing to submit information about samples relating to unpublished research to a web service that is managed by an unaffiliated research group. This constraint can make users uncomfortable as there is no way they can verify that their unpublished data are not being stored indefinitely on a server outside their control. The Excel application Cosmocalc

3. `hess.ess.washington.edu/math` and `www.cronuscalculators.nmt.edu`

(`cosmocalc.googlepages.com`) can be downloaded locally, which avoids that privacy issue. However, geoscientists are then stuck with the capabilities of Excel, and the limitations of using spreadsheets for this type of software was a large part of the motivation for the ACE project in the first place. ACE matches and exceeds the functionality provided by all of the tools mentioned in this paragraph; it handles all of their isotopes and calculations, and more. Its components and workflows incorporate a wide array of current research in cosmogenic dating (Desilets & Zreda, 2003; Lifton et al., 2005; Pigati et al., 2008). Indeed, it is ACE's support for extensibility that allows it to offer features and services that are not found in any other cosmogenic dating software package.

Our work on ACE is closely associated with research on design environments (Winograd, 1995) and domain-specific software architectures (Tracz, 1995). Design environments are characterized not only by functionality to perform a particular task, but also by the services they provide to help users reflect on the task itself. ACE, for example, not only calculates ages based on measured inventories of cosmogenic nuclides. It also provides functionality to analyze and compare the results of cosmogenic dating techniques, which allows researchers in this domain both to refine existing techniques and to design new ones. Design environments have also been used to help users understand and apply standardized techniques within a domain. For instance, (Garlan, Allen, & Ockerbloom, 1994) developed techniques for customizing software architecture design environments with information about particular software architectural styles. These customizations allowed a user to more quickly create software systems that followed these styles. In ACE, we have carefully designed the ways in which users can extend the design environment, allowing them to add new data sets, new components, new scaling factors, and new plot types as their needs evolve. These standardized extension points will allow experts in this problem domain to teach novices (e.g. graduate students) the ways in which cosmogenic dating techniques can be altered and evolved. None of the other software systems for cosmogenic dating offer this kind of flexibility, extensibility, and support for learning.

Within the field of argumentation as a reasoning framework, two major branches have been established (Reed & Grasso, 2007). One branch views argumentation systems from a dialectical viewpoint, largely based on (Dung, 1995). Systems using this variety of argumentation include (Hunter, 2004) and (Prakken, 1996). The other branch approaches argumentation as an extension to and improvement on first-order logic, and is largely based on the Logic of Argumentation (LA) of Krause et al. (Krause et al., 1995). Systems following this tradition include (Cayrol & Lagasquie-Schiex, 2003; Morge & Mancarella, 2007). Calvin draws on both of these branches of argumentation: primarily, its rules and confidence system are drawn from LA, but the perspective of treating arguments as trees and the method of performing first a local and then a global argument assessment are more common in the dialectical branch. Calvin's definition and use of confidence was also informed by (Farley, 1997) and (Amgoud, Bonnefon, & Prade, 2005).

Although argumentation is the variety of reasoning that best matches the structure of expert reasoning and communication in this domain, there are many kinds of non-monotonic logics (designed specifically around handling contradiction) that we considered for Calvin's design. These include circumscription (McCarthy, 1980; McCarthy, 1986), default reasoning (Reiter, 1980), and other forms of nonmonotonic reasoning (Gaines, 1996; Pereira, Alferes, & Apar'icio, 1991; Pollock, 1994). While each of these is useful for many problems, none of them solves all of the issues involved in cosmogenic isotope dating analysis. In particular, most varieties of non-monotonic reasoning—besides argumentation—require explicit definitions of when conclusions may be withdrawn and have problems with the issues of partial support and defeat. Furthermore, many nonmonotonic logics are unsound in some way, making their use problematic (Etherington, Kraus, & Perlis, 1991). These properties made these logics unsuitable for CalvinâĂŹs framework.

It is possible to view cosmogenic isotope reasoning as a diagnostic process. Model-based diagnosis, as in (Lucas, 1997; Santos, 1991; Struss, 2004), is inappropriate for Calvin because complete models of most geologic processes simply do not exist. Non model-based diagnosis systems do exist, including systems that handle contradiction (again, a major requirement for Calvin), for example (Doyle, 1983; Gaines, 1996). However, these diagnosis architectures use "absolute" rules, which are not appropriate in Calvin's application.

5. Conclusion

ACE represents a significant advance in the capabilities provided by cosmogenic dating software. It provides geoscientists with the ability to both evaluate existing dating techniques and design new ones. It is based on a comprehensive conceptual framework that ensures that all aspects of cosmogenic dating techniques are modeled by the system. Its software architecture provides both flexibility and extensibility to allow researchers to explore "what if" scenarios on how these techniques can be evolved. Its reasoning engine captures and operationalizes the knowledge of more than two dozen experts in the field, allowing it to work through scientific scenarios automatically, saving its user from the time and aggravation of an exhaustive exploration of the hypothesis space—and occasionally finding explanations that they have missed.

ACE's extensive and long-term use by the forensic paleoclimatology research community is a testament to the success of our goals in building it. Indeed, the system remains in active use to this day, several years after the development cycle ended, without a single request for help from the geochemists to the computer science side of the team. As mentioned in the abstract, the ACE project website has received over 17,000 hits since 2008, including 2500 over the last twelve months. The software has been downloaded 589 times as of April 2013, which is a significant number in a research community of $O(10^2)$ PI-level scientists. Given the state of cosmogenic nuclide dating software prior to the development of ACE, we believe that our contributions have enabled new science, allowing progress to be made in this domain that previously was hindered by inflexible computational tools that offered no automated reasoning assistance[4].

In the last fifteen years, agencies that fund computer science research have placed a strong emphasis on the need for multidisciplinary research. Programs such as the National Science Foundation's Information Technology Research for National Priorities (ITR) (Zorn et al., 2004) and Cyber-Enabled Discovery and Innovation (Misawa, Russell, & Whang, 2008) programs typically funded large, multidisciplinary projects that advanced computer science by having computer scientists work with scientists from other disciplines providing benefits to all. We hope that ACE stands as evidence of the success of these programs. We also hope that our approach to working as an

4. In the past year, another automated reasoning tool for cosmogenic isotope dating has been released (Applegate et al., 2012), but that tool works only for the specific case of moraines.

interdisciplinary team, as captured in this paper, stands as a useful model to future teams working in this context. Its benefits are indeed those touted for multidisciplinary computer science research: the domain scientists gain tools that greatly enhance their ability to perform research in their own fields; in responding to the demands of the domain scientists, the computer scientists make advances in their respective areas of specialization.

The computer scientists in the ACE team have now moved on to applying this multidisciplinary research approach to a different domain: the analysis of ice and ocean-sediment cores. From these cores—depth-wise sequences of information—a geoscientist interested in a past climate event must first deduce the timeline for the data: that is, a curve called an *age model* that relates the depth in the core to the age of the material at that point. This is the first critical step in reasoning about the science of the events that produced the core. Like ACE, this new project (entitled CSCIENCE) brings together computer scientists and geoscientists around the goal of producing a software system that enables scientific progress in a challenging application domain. There are many challenges in the CSCIENCE project, some familiar from the previous pages (current tools and data sets are stored in Excel spreadsheets) and some requiring fundamental new work in the areas of big data (storage and processing) and automated reasoning. Assumptions about how ice and ocean-sediment cores are created have multiple permutations, for instance, leading to a potential explosion in the number of age models to generate and evaluate. And the data involved are very different. ACE's reasoning engine worked with two numbers (mean and standard deviation) for each of a few dozen rock samples taken from a single landform that was formed instantaneously in geological time, then influenced by a small list of candidate processes that involved no unknown parameters. CSCIENCE's data sets are thousands or millions of times larger, and their ordered nature allows reasoning about *continuous* events, not just episodic ones, which is a much harder and more general problem.

We believe the lessons learned in designing, building, deploying, and evaluating ACE —in terms of project management and multidisciplinary research, as well as the scientific contributions to software engineering and artificial intelligence—will serve us well in tacking this complex new application domain.

Acknowledgements

This material is based upon work sponsored by the NSF under Grants ATM-0325812 and ATM-0325929.

References

Amgoud, L., Bonnefon, J., & Prade, H. (2005). An argumentation-based approach to multiple criteria decision. *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*.

Anderson, K., Bradley, E., Rassbach, L., Zweck, C., & Zreda, M. (2010). End-to-end support for paleolandform dating. In N. Adams, M. Berthold, & P. Cohen (Eds.), *Advances in intelligent data analysis ix*, Vol. 6065. Springer Lecture Notes in Computer Science. Proceedings of the 9th International Symposium on Intelligent Data Analysis (IDA).

Anderson, K., Bradley, E., Zreda, M., Rassbach, L., Zweck, C., & Sheehan, E. (2007). ACE: Age calculation engine: A design environment for cosmogenic dating techniques. *Proceedings of the International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP).*

Applegate, P., Urbana, N., Kellera, K., Lowell, R., Laabs, B., Kelly, M., & Alley, R. (2012). Improved moraine age interpretations through explicit matching of geomorphic process models to cosmogenic nuclide measurements from single landforms. *Quaternary Research*, *77*, 293–304.

Ballantyne, C., Stone, J., & Fifield, L. (1998). Cosmogenic Cl-36 dating of postglacial landsliding at The Storr, Isle of Skye, Scotland. *The Holocene*, *8*, 347–351.

Cayrol, C., & Lagasquie-Schiex, M.-C. (2003). Gradual acceptability in argumentation systems. *Proceedings of the Third International workshop on computational models of natural argument*.

Chamberlain, T. (1965). The method of multiple working hypotheses. *Science*, *148*, 754–759. Reprint of 1890 *Science* article.

Desilets, D., & Zreda, M. (2003). Spatial and temporal distribution of secondary cosmic-ray nucleon intensities and applications to in-situ cosmogenic dating. *Earth and Planetary Science Letters*, *206*, 21–42.

Doyle, J. (1983). Methodological simplicity in expert system construction: The case of judgments and reasoned assumptions. *AI Magazine*, *4*, 39–43.

Dung, P. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n-person games. *Artificial Intelligence*, *77*, 321–357.

Etherington, D., Kraus, S., & Perlis, D. (1991). Nonmonotonicity and the scope of reasoning. *Artificial Intelligence*, *52*, 221–261.

Farley, A. (1997). Qualitative argumentation. *Proceedings of the 11th International Workshop on Qualitative Reasoning (QR)*.

Fischer, G., McCall, R., Ostwald, J., Reeves, B., & Shipman, F. (1994). Seeding, evolutionary growth and reseeding: Supporting the incremental development of design environments. *Proceedings of the SIGCHI Conference on Human factors in Computing Systems* (pp. 292–298).

Gaines, B. (1996). Transforming rules and trees into comprehensible knowledge structures. *Advances in Knowledge Discovery and Data Mining* (pp. 205–226).

Garlan, D., Allen, R., & Ockerbloom, J. (1994). Exploiting style in architectural design environments. *Proceedings of the 2nd ACM SIGSOFT Symposium on Foundations of Software Engineering* (pp. 175–188).

Hunter, A. (2004). Making argumentation more believable. *Proceedings of the Nineteenth Conference on Artificial Intelligence (AAAI)* (pp. 269–274).

Jackson, L., Phillips, F., Shimamura, K., & Little, E. (1997). Cosmogenic 36Cl dating of the Foothills erratics train, Alberta, Canada. *Geology*, *25*, 195–198.

Krause, P., Ambler, S., Elvang-Goransson, M., & Fox, J. (1995). A logic of argumentation for reasoning under uncertainty. *Computational Intelligence*, *11*, 113–131.

Lifton, N., Bieber, J., Clem, J., Duldig, M., Evenson, P., Humble, J., & Pyle, R. (2005). Addressing solar modulation and long-term uncertainties in scaling secondary cosmic rays for in situ cosmogenic nuclide applications. *Earth and Planetary Science Letters*, *239*, 140–161.

Lucas, P. (1997). Symbolic diagnosis and its formalisation. *The Knowledge Engineering Review*, *12*, 109–146.

McCarthy, J. (1980). Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, *13*, 27–39, 171–172.

McCarthy, J. (1986). Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, *26*, 89–116.

Misawa, E., Russell, T., & Whang, K. (2008). Cyber-enabled discovery and innovation (CDI). `www.nsf.gov/funding/pgm_summ.jsp?pims_id=503163`.

Morge, M., & Mancarella, P. (2007). The hedgehog and the fox: An argumentation-based decision support system. *Proceedings of the 4th International Workshop on Argumentation in Multi-Agent Systems*.

Pereira, L., Alferes, J., & Apar'icio, J. (1991). Nonmonotonic reasoning with well founded semantics. *Proceedings of the Eighth International Logic Programming Conference* (pp. 475–489).

Phillips, F. (2012). Cronus-earth project: Results. *Quaternary International*, *279–280*, 379. http://dx.doi.org/10.1016/j.quaint.2012.08.1175.

Pigati, J., Zreda, M., Zweck, C., Almasi, P., Elmore, D., & Sharp, W. (2008). Ages and inferred causes of Late Pleistocene glaciations on Mauna Kea, Hawai'i. *Journal of Quaternary Science*, *23*, 683–702.

Pollock, J. (1994). Justification and defeat. *Artificial Intelligence*, *67*, 377–407.

Prakken, H. (1996). Dialectical proof theory for defeasible argumentation with defeasible priorities. *Proceedings of the Biannual International Conference on Formal and Applied Practical Reasoning Workshop* (pp. 202–215).

Rassbach, L. (2009). *Calvin: Producing expert arguments about geological history*. Doctoral dissertation, University of Colorado.

Rassbach, L., Anderson, K., & Bradley, E. (2010). Providing decision support for cosmogenic isotope dating. *Proceedings of the 22nd Conference on Innovative Appliations of Artificial Intelligence (IAAI)*.

Rassbach, L., Anderson, K., & Bradley, E. (2011). Providing decision support for cosmogenic isotope dating. *AI Magazine*, *32*, 69–78.

Rassbach, L., & Bradley, E. (2008). Challenges in presenting argumentation results. *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Rassbach, L., Bradley, E., Anderson, K., Zreda, M., & Zweck, C. (2007). Arguing about radioisotope dating. *Proceedings of the International Workshop on Qualitative Reasoning about Physical Systems*.

Reed, C., & Grasso, F. (2007). Recent advances in computational models of argument. *International Journal of Intelligent Systems*, *22*, 1–15.

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, *13*, 81–132.

Santos, E. (1991). On the generation of alternative explanations with implications for belief revision. *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence (UAI)* (pp. 339–347).

Stenning, K., & van Lambalgen, M. (2008). *Human reasoning and cognitive science*. MIT Press.

Struss, P. (2004). Deviation models revisited. *Proceedings of the Eighteenth International Workshop on Qualitative Reasoning*.

Tracz, W. (1995). DSSA (domain-specific software architecture): Pedagogical example. *ACM SIGSOFT Software Engineering Notes*, *20*, 49–62.

Winograd, T. (1995). From programming environments to environments for designing. *Communications of the ACM*, *38*, 65–74.

Zorn, M., Cherniavsky, J., Suskin, M., Papitashvili, V., Iacono, C., Meacham, S., Kaper, H., & Newlon, D. (2004). Information technology research for national priorities (ITR). `http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=5524`.

Zreda, M., Bradley, E., & Anderson, K. (2003). Software for interpretation of cosmogenic isotope inventories - combination of geology, modeling, software engineering and artificial intelligence. NSF Award Numbers 0325812 & 0325929.

Zweck, C., Zreda, M., Anderson, K., & Bradley, E. (2012). The theoretical basis for ACE, an age calculation engine for cosmogenic nuclides. *Chemical Geology*, *291*, 199–205. doi:10.1016/j.chemgeo.2011.10.005.