Enhancing Team Projects Using an Event-Triggered Knowledge Network

Jeff DePree*

Department of Computer & Information Science & Engineering
University of Florida, Gainesville, FL, USA
E-mail: jdepree@cise.ufl.edu

Xuelian Xiao

Department of Computer & Information Science & Engineering
University of Florida, Gainesville, FL, USA
E-mail: xxiao@cise.ufl.edu

Stanley Y. W. Su

Department of Computer & Information Science & Engineering
University of Florida, Gainesville, FL, USA
E-mail: su@cise.ufl.edu

*Corresponding author

Abstract: Across the disciplines and levels of education, team projects are becoming an increasingly important tool for assessment. However, it is often difficult to make sure that work is fairly distributed, that a reasonable schedule is formed which does not leave everything for the last minute, and that individual students are fairly rewarded/penalized for their respective contributions. To solve these problems, we can model a project using events, rules, and workflows. Depending on the nature of an event, an appropriate rule can be triggered, which can subsequently initiate a workflow that will assign specific tasks to specific roles within the team. Events that occur over the course of the project, fired from a variety of sources, can lead to the derivation of new knowledge, and potentially alter the flow of the team's activities. An integration of mobile devices into the system can insure that students are always aware of the current state of the system and their roles within it. At the conclusion and at all prior points in the project lifecycle, a comprehensive log of each student's activities will be available and will greatly simplify the task of assigning fair and accurate grades. The result will be a more educational, more equitable, and far more engaging learning experience.

Keywords: Collaborative Learning; Event-Triggered; Rule Processing; Workflow Learning; m-Learning

Biographical notes: Jeff DePree is a PhD student in the Department of Computer and Information Science and Engineering at the University of Florida. He received his B.S. in Computer Engineering from the University of Florida in 2006. He is interested in technologies for ubiquitous collaboration to improve access to education and healthcare.

Xuelian Xiao is a PhD student in the Computer and Information Science and

Engineering department at the University of Florida. She received her B.S. in computer science and M.E. in computer engineering from Sun Yat-sen University, Guangzhou, China, in 2000 and 2003, respectively. Her research interests include knowledge management, E-business, distributed systems and rule-based systems.

Dr. Stanley Y. W. Su is Distinguished Professor Emeritus and Adjunct Professor of the Department of Computer and Information Science and Engineering at the University of Florida. He received his M.S. and Ph.D. degrees in Computer Science from the University of Wisconsin, Madison, in 1965 and 1968, respectively. He is an IEEE Fellow.

1. Introduction

Collaborative learning is a teaching method that has been used effectively in classrooms from kindergarten to college, and in every subject area. It has been shown to provide a host of benefits, improving students' skills in everything from conflict management to handling cultural diversity (Gillies, 2007). The method covers a large spectrum of learning activities, from two students studying for a test together, to thousands of students around the globe utilizing e-learning software to complete an online class (Dillenbourg, 1999). This paper will focus on a specific point in that range where a group of students, in the context of a larger class or community, works together to solve a common problem; this is typically labeled as a team project.

In an educational setting, team projects have many of the same advantages and present many of the same challenges that are seen in business scenarios. Theoretically, adding more people to a team should significantly increase the amount of the work the team can complete in a given period of time. However, without proper management, having multiple contributors on a project can easily become more of a hindrance than a help. With a single deliverable, often not divisible in the least, and no fixed schedule, it can become very difficult to insure that all team members do a similar amount of work, and that they do it in a timeframe that allows for verification and integration with the other components.

Because students' mediation and scheduling skills, and their motivation to apply them in an academic setting, are often lacking, and interaction with the instructor is limited, projects like this stand to benefit a great deal from software support. The field of computer-supported collaborative learning studies how technology can be applied to assist students in their collaborative efforts. Such software does not deliver instruction itself, but rather provides a medium for communications among the students, as well as a structure for that communication (Stahl, Koschmann, & Suthers, 2006). This paper will examine how a particular set of software tools, the Event-Triggered Knowledge Network (ETKnet) system, can be used to enhance the ability of a team to manage their project, and to keep an accurate record of the work performed by the different members of the team.

To form a better impression of why such tools are needed, the following sections will use a running example of a particular team project where their use is especially apropos; it should be noted that this example represents just a single instance for which these tools are applicable, and that they can be used in a very broad range of subjects and types of projects. This assignment is one that is given each semester as the central

component for the undergraduate software engineering class at the University of Florida. The class is broken into groups of 5-15 students and each group is tasked with selecting an open source software project to improve upon over the course of the semester. Each group decides on a management structure and process model, specifies deliverables, and sets a detailed schedule, and then attempts to adhere to this framework as they progress through the semester and become more familiar with the task at hand. The teams are assessed not only on their contributions to the open source software, but also on how well they follow the structure they put in place (Dobbins, 2007).

Obviously, there is a lot of potential for a project such as this to gain from a mediating software tool. It would be extremely useful if there were an automated means of fairly dividing the work among the team members and logging who accomplished which tasks in what amount of time. These capabilities can be realized by applying techniques from the areas of workflow learning and rule-based systems.

Workflow learning borrows the idea of workflow management from the business domain and adapts it to an academic environment. A number of roles are defined, each of which is responsible for carrying out specific tasks. These tasks are assembled into a process graph; tasks may occur sequentially or concurrently and may rely on outputs from earlier tasks or contribute inputs to later ones. One or more students are assigned to each role and are required to complete the associated tasks (Lin, Ho, Orlowska, & Sadiq, 2002). This concept has been used to model courses as a whole, as in Virtual Campus (Cesarini, Monga, & Tedesco, 2004) and Flex-el (Lin et al., 2002), where a single workflow may be stretched over an entire semester and mediate every facet of the course. The aim is to achieve a middle ground between the two extremes of having each student produce his/her own results, and having a group of students cooperate to produce a single result; while the tasks may all work toward a single end product, every step of the process is supervised and every participant is held accountable (Martel, Vignollet, Ferraris, David, & Lejeune, 2006).

Rule-based systems capture human knowledge and apply it in an automated decision-making process. The system uses a knowledge base of if-then rules to move from the question to an answer; the rules are not explicitly structured, but are chosen by comparing their antecedents with the current state of the system (Hayes-Roth, 1985). One common use of these systems in the learning domain is the "e-tutor" which uses rules to recommend to the student an appropriate course of action for a given situation (Odeh & Ketanah, 2007). Carro, Ortigosa, and Schlichter suggest a system that varies its interface based on a given student's specific needs; for example, if the student is a visual learner, he/she might be presented with graphical learning tools, whereas if he/she were a textual learner, the interface would be largely text-based (2003). Tsai and Tseng propose a system where the material itself may be swapped out on the basis of the results of earlier evaluations (2002).

The rest of the paper is divided up into the following sections: Section 2 discusses event and rule processing and how ETKnet fits into this space, Section 3 goes into more depth on how ETKnet can be applied to the collaborative learning domain, Section 4 describes how all the details of the students' efforts are recorded for assessment by the instructor, Section 5 explains the process of setting up the structure of a project and subsequently carrying out the project using a web-based interface, Section 6 tells how the use of cell phones and other mobile devices can increase the effectiveness of the system, and finally, Section 7 summarizes the system and provides some final thoughts on its potential utility.

2. Event and Rule Processing

When communicating between two or more entities, it is necessary to establish some protocol so that the messages transmitted by a given entity can be understood by the other participants. In traditional protocols, it is necessary to specify one or more recipients for a given message, and then to insure that said recipients are devoting their attention to receiving at the same time that the sender sends. This approach, while effective in many scenarios, is not appropriate for situations in which the identities of the interested parties are not obvious, or in those where the time at which the sender will send cannot readily be determined. In these situations, a better tack is to use the publish/subscribe protocol; using this protocol, interested parties register their interest (subscribe) for a given class of information from a given source, and when that source has new information, it sends (publishes) it to all registered entities using the addresses they have provided (Sahingoz & Erdogan, 2003).

A publisher can generate a great deal of information and not all of it may be relevant to a subscriber. There must therefore be a way for each subscriber to define filters that will make it possible to ignore all delivered data except for that which is useful to the specific task at hand, and then to take action based on that data. This can be accomplished using rules, which apply logical expressions to each piece of sent data to determine whether or not it should be forwarded to the subscriber's applications. Unlike most traditional computer programs, these rule-based systems allow for new knowledge, in the form of additional rules, to be inserted incrementally and with minimal syntax; virtually all classes of users, not just computer programmers, are capable of understanding existing rules and adding their own (Hayes-Roth, 1985).

Rule-based systems typically make use of one or more of the following rule types: integrity constraints, derivation rules, and action-oriented rules. An integrity constraint checks whether a given field in a data set is within appropriate bounds or has the appropriate relationship to other fields in the set. A derivation rule infers new data from that which is already available in the system. Finally, an action-oriented rule can contain a workflow specification in the action clause of the rule and executes the specified workflow, which may involve multiple automatic and manual operations, when the event data satisfies a given condition (Xiao, DePree, Degwekar, Su, & Beck, 2008).

ETKnet, or Event-Triggered Knowledge Network, is an implemented rule-based system that utilizes all three of these rule types to share multi-faceted knowledge among a network of loosely-coupled collaborative sites (Degwekar, DePree, Beck, Thomas, & Su, 2007a; Xiao et al., 2008). Collaborating entities can define events and rules at their own sites using ETKnet's user interface tools. Workflow processes are specified in the action part and the alternative action part of an action-oriented rule. An event, which has occurred at one site, can trigger a rule at another site, or different rules at multiple sites, which can each in turn initiate a workflow that can carry out operations across multiple sites. All types of rules are translated and packaged as web services so that they can be processed uniformly anywhere, without the need for separate workflow processing and rule evaluation engines.

Figure 1 provides a more detailed view of the architecture of the system. Each collaborating site in the federation contains the software tools needed to define events, rules and workflows, as well as a rule server and an event server capable of processing them. One of the sites also serves as the host site and is responsible for coordinating communication among the collaborating sites, as well as maintaining a registry of web services for discovery and use by those sites. When an event occurs at one site, its event

server notifies the event servers at all subscriber sites, which subsequently signal for the corresponding rule servers to process the appropriate rules and workflows.

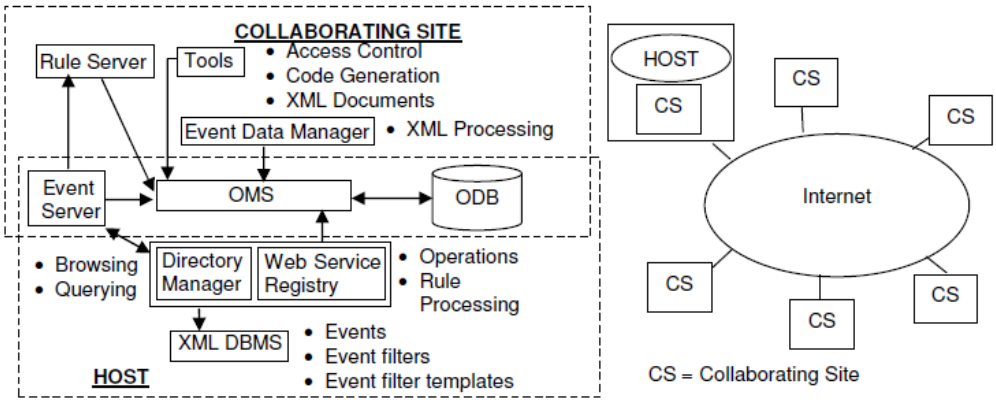

Figure 1. Architecture of the collaboration federation

To further explain the functionality of ETKnet, let us look at a basic, yet illustrative example. For this example, let us assume that a class of thirty students is broken into groups of ten students each; all of these students would register with the websites corresponding to their respective groups and the instructor is registered with his/her own site. The instructor has installed rules at each group's site that will be triggered by a project event and initiate a workflow to tell the students what to do. With this structure in place, the instructor logs into his/her site and posts an event to begin a team project. The event triggers the appropriate rule on each of the three sites, and starts a workflow on each. The first operation in each workflow is associated with one of the students in each group, and that student receives an SMS on his/her phone that gives him/her instructions on what needs to be done. The student completes the requested task, logs into his/her group's site and types up the results; this causes the workflow to move to the next step and supply a notification, with the results from the first step, to a different student. It is also possible that several steps in the workflow may occur concurrently; in this case, multiple students would be notified with their respective tasks. When the workflow completes, an event can be fired that will send the collected results to the instructor and notify him/her that the project has been finished.

The example of the previous paragraph is very straight-forward, but one can imagine many variants. It is possible that one of the students might log in and fire his/her own event, which might cause a rule within one or both of the other sites to be triggered; this might happen if, for example, one group reached a critical point before the other groups and the other groups' workflows needed to be altered as a result. Alternatively, you might have a rule on one site that is triggered when certain conditions evaluate to true on each of the other two sites. The possibilities are only limited by the rules put in place by the instructor and/or students.

3. Application of an Event-and-Rule-Based System in Collaborative Learning

It is not difficult to see how ETKnet may be readily applied to the academic domain to motivate collaborative learning. In the instance of a team project, each team can be represented by a site in the system's network, which can participate in one or more

collaborations. At the most obvious level, the team belongs to a class and receives events from the instructor, as well as events from other teams in the class who may be working on related aspects of the project. It can be left up to the team to determine what rules to define to react to the events which are applicable to its approach. A given team might additionally form collaborations with other teams that are producing components that depend upon or are dependent upon the components for which the team is responsible. The rules defined by the team can activate workflows in response to events; these workflows cause a series of operations to be assigned to different roles, which are filled by the students within the group.

The following sub-sections will take a closer look at the different components of this application, particularly as they apply to the software engineering example discussed earlier. We will focus on a specific scenario where the source code is updated by a third party and it becomes necessary for the team to download the new code, merge any changes with those that have been made by the team itself, and adapt to the new state of the system.

3.1. Notifying system and participants using events

The system's execution is driven by events. These events signify changes in the state of the system or the environment, which necessitate changes to the requirements of the assignment, or in the information made available to the students or instructor. At the most basic level, these include requirement specification by the instructor and submissions by the students, but the idea can be extended to accommodate unanticipated student contributions, or notifications by autonomous agents that poll sensors or web resources at regular intervals.

When an instructor or teaching assistant wants to initiate a new requirement for the assignment, he/she needs only to post an event with the appropriate text and attachments that will cause the appropriate students to be contacted with the data necessary to complete the assignment. A nearly identical process can be employed when it is desired to update an existing assignment with new requirements or new information. In the case of the software engineering example, the instructor might remove, add, or reorder requirements to simulate the changing demands of a software customer; this might also be done as a sort of load balancing to force the team to reallocate the bulk of the workload from one set of students to another.

In a similar vein, events can be used to collect both the intermediate and final results of the students' work. When a student has finished his/her part of the assignment, or a subpart thereof, he/she can post an event with the results of his/her work which will cause both the instructor and the other students to be notified. The current state of the system will be updated to reflect the completion of the task and anticipate the next step in the process. Our example would require that a set of deliverables be provided by each team at regular intervals, and these deliverables could be packaged as event data, which are data associated with a given occurrence of an event.

Beyond these fundamental elements, the system will also allow for events originating from students which do not pertain directly to task completion, but may provide useful information for other students or for instructor audits. These events will be tagged with keywords or other labels that will allow them to be picked up by interested parties. These parties may be passively searching for a solution to a given problem or may be assigned a task in which they are likely to encounter a problem for which a solution can be provided. Furthermore, the data from such events may be analyzed along

with that from other events, in conjunction with the current state of the system, to infer new knowledge and take actions based on that knowledge. In our example, the various teams are working on different projects, but techniques or modular software components employed within those projects may be the same, and thus it would be advantageous for one team to communicate a useful discovery to members of other teams that may be battling the same problem.

A final type of event could originate from sensors in the environment. If a student were assigned a weather-dependent task, barometric sensors could alert him/her that he/she can proceed with the task. One could envision similar scenarios with tasks involving human or animal activity, vehicular traffic, accesses to an Internet resource, or anything else where asynchronous updates would be useful. In every case, the potential far exceeds simple notification, as sensor values may be aggregated to draw conclusions based on a perceived global state. When dealing with open source projects as in our example, it is very useful to know when other users, apart from those in the class, make contributions to the project, or at a higher level, a new version of the software is released; the knowledge of such an event makes it possible for the team to restructure its efforts based on the new state of the software, and not duplicate work that has already been completed by somebody else.

In our scenario, a developer external to the team adds a feature to the software and uploads the new versions of the source files she changed. The team has defined an event which is activated by a service that periodically checks the project website for modifications within certain fields. When the system detects that the content of the page has been changed, an instance of the event is fired for each code upload, and this event contains as its event data the time of the change, the identity of the user, a new version number for the software, and links to the files that were changed.

3.2. Analyzing data and inferring new knowledge with rules

Rules serve to evaluate and act upon information provide by events. Based on the content of event data, rules may be chosen that can issue notifications to students and instructors, effect workflows that dictate the distribution and sequence of tasks, and infer new knowledge that can, in turn, allow other rules to be evaluated.

Given the traditional structure of an assignment, an instructor would define a rule that responded to a requirement specification event by initiating a workflow that would define which students carried out what tasks along with the deadlines for those tasks. Additional rules would handle submission events posted by students, mid-project changes in the specifications, hints posted by students for the benefit of their classmates, and any external stimuli that might be relevant to the project.

In a more advanced class, the students could be solely responsible for defining the necessary rules to allow for an optimal response to requirement changes, environmental factors, and input from other students. These responses to expected and unexpected events, along with the derivation of new knowledge relevant to the project could be important components of the problem-solving process. This would be the case in our example; since the instructor would have little or no knowledge of the open source project that was chosen, it would be necessary for the team itself to define an appropriate workflow, as well as rules that would react to events appropriately based on the team's specific requirements.

In our scenario, a rule would be defined which would react to new versions of the source code by executing a workflow. This rule checks to see if a new version number is

equal to one more than that of the current version of the code; since this new version number is a subset of the event data provided by the event described above, an implicit trigger is established between the event and the rule, and whenever the event is fired, the rule is evaluated. If multiple instances of the event were fired at the same time, the rule would be evaluated for each of the events, but would initially only effect a workflow for the event with the lowest version number; once the appropriate action were taken for that event, the rule would be evaluated for each of the other event instances using an updated version number.

3.3. Using workflows to distribute tasks and track completion

At the heart of any assignment would be one or more workflows which would distribute the required tasks among the students, transfer the results of the tasks from one student to the next, and track the completion of the assignment.

Each task within a workflow has associated with it inputs, outputs, and a role responsible for its completion. When the firing of an event or the completion of previous tasks allows the system state to reach a given task, the inputs for the task are taken from event data and the outputs of previous tasks, and are sent along with a notification to any student(s) who is(are) subscribed to the specified role. When a student has completed his/her assigned task, he/she will provide the requested outputs and the system will move to the next step in the workflow. In our software example, those students filling the testing role will need to wait for those filling the developer role to finish the current iteration of the code; the workflow would accept the code from the developers and transfer it to the testers, which would be responsible for the next operation in the sequence.

A workflow can be composed of any combination of a number of different constructs. At the most basic level, operations can be executed sequentially, but parallel processing of operations is also possible. A flow can move from sequential to parallel by way of a 'split' construct, where the parallel operations can either execute unconditionally following the completion of the preceding operation, or their execution can be contingent on the evaluation of conditional statements. Likewise, parallel threads of execution can merge back into a sequence by way of a 'join' construct; here, the next operation can begin its execution as soon as a given number of the parallel operations complete. If order is not important, operations can be added to an unordered collection and executed in whichever order is most convenient. If it is necessary to execute a single operation, or sequence of operations, multiple times, then a looping construct can be used (Xiao et al., 2008).

A given team is not limited to the operations that have been defined within that team. Operations that have been defined by the instructor, or by other teams in the class, can also be incorporated into the team's workflows. Assuming that an operation is an automatic web service, or it is a manual operation which includes roles to which students from the team in question are assigned, it is a simple matter for that team to adopt it with no modifications. An instructor may provide a pool of pre-made operations that each team may select and use as the basis of its workflows. But if this is not done, and the teams each define their own operations, then a team might identify a parallel between the tasks that it must perform and those that another team must perform and set up a collaborative relationship with that team to share its operations accordingly.

For our scenario, a workflow would be defined that contains three manual operations, which are assigned to three separate roles within the team. The first operation

is assigned to the repository manager role, and it requires a student to download the changed files and update the team's repository. Once the student has informed the system that this action has been completed, the coding team will be informed of the change, and one student within that subgroup will be responsible for integrating the new changes into the existing source and insuring that the code compiles and satisfies some basic functionality tests. When this is done, the testing team will be notified and one tester will need to thoroughly test the new version of the software; the tester will report back to the system upon the completion of the tests. Depending on the outcome of the tests, the workflow will either finish, or transfer control back to the previous operation so that the coding team can fix any new bugs that were found. For each operation, every student within a given role will receive the notification, but only the person who first responds to the notification will be responsible for the completion of the operation. Data from each step, such as any bugs that are discovered, will be provided by the appropriate student and attached to the notifications for future steps.

4. Audits

It is sometimes the case that instructors assign team projects with the expressed goal of avoiding the extra effort of grading individual performance, but more frequently, this is just a natural side effect. In the vast majority of group work, instructors prefer to adopt the naïve assumption that the results of the whole are representative of the component parts, without making any attempt to delve deeper to figure out whether the workload was, in fact, equably distributed. This often results in a portion of the students doing little or no work, while the remainder must put in extra hours to compensate. Even if an evaluator does try to ascertain who did what, individualized paper trails are typically non-existent, and students are usually reluctant to give any information to implicate their teammates.

The system described in this paper eliminates this problem by providing a detailed log of the entire execution of the project, including all the activities performed by each student. The system records when each operation is assigned and to whom, as well as the completion time of that operation and the corresponding results. This makes it immediately clear what contributions each student made and the time it took him/her to complete each task. An automatic grading system could be devised to analyze this information and assign scores based on a number of criteria, but in most cases, an instructor would use any anomalies within this data as a jumping-off point for further investigation.

In the case where the students themselves create the events, rules, workflows and other components, the structure itself could be used as a basis for assessment. If teams within the same class design projects of different complexity, then the instructor should take the scope of each project relative to that of the others into consideration so that fair scores can be assigned to members of different teams. The use of rules to react to events, both fired from within the group and from elsewhere in the system, would be a telling metric for how well the group understands its role in the environment. The division of tasks in the group's workflows, both from the standpoint of how much each student assumes, and how efficiently and logically the work can be carried out, will be another means of evaluation. In upper-level courses, the management of the work can become just as important, if not more important than the deliverables themselves.

A log entry will be sent from each team's site to the instructor's site whenever an event is fired, a rule is evaluated, or an operation is completed. By default these will be

grouped by team, but it will also be possible to join all of the team logs together to get a global view of the system. Optionally, this global view may be made available to the students so that they may gauge their performance and interactions with respect to the rest of the class. The elements of this view can be aggregated into student-specific and team-specific summaries that may give some impression of individual and team performance relative to the other students in the class. These summaries might include the response time and correctness of outputs for completed tasks relative to class averages or instructor expectations, or they may provide the degree to which the students captured pertinent data from events and leveraged existing resources in the collaborative network. Clearly, for cases in which the students define their own projects, the utility of such summaries is limited and the instructor will be required to take a more active role in the feedback process.

5. User Interface

The user interface consists of two main parts. The first is used for the definition of the components that drive the flow of a project; this interface is typically used by the instructor, but could also be used in a scenario where students are required to define their own project structure. This latter option would be the most appropriate for our software engineering scenario. The second part is used by the students to fire manual events, report on the completion of operations, and view the state of the system. Both parts are web-based and use a thin-client architecture; the entire interface is viewable in a normal web browser without the need to download any additional software.

Before a given team can interact with the rest of the class, it must join the class's federation. In a controlled lab, the instructor may add each of the teams' servers to the federation, but in the case that students are using their own machines, the instructor may make the location of the host server known and each team can register their machine with the host as part of an installation process. Once the servers have been registered, the individuals on each team must register with the system; again, they can either be added by the instructor or add themselves using a registration wizard. Depending on the nature of the assignment, the instructor may define a set of global roles that each team is expected to apply to its members, or, alternatively, each team can define its own roles to fit a team structure that it defines.

The system supports the definition of manual and automatic events. Manual events require that a label, description, and data fields be defined so that students are made aware of the intent and the required content of each event; a set of roles, describing which students are allowed to fire a given event, is also supplied. Automatic events can either monitor a website or a sensor. For the website case, a web address, a field in the target website, and a read interval would be specified; the system will periodically load this web site to look for changes in the selected field. In the case of the sensor, an address for the sensor, or specific sensor value, and a read interval will be given so that the system can regularly pull data from the sensor.

To make it possible to define arbitrarily complex rules, the system provides an interface that expands according to user selections. The user decides which type of rule to define, and the system then guides the user through the specification of each part needed for the rule. Drop-down menus, auto-complete, help screens and error checking allow the user to consider a large range of possibilities with little or no training.

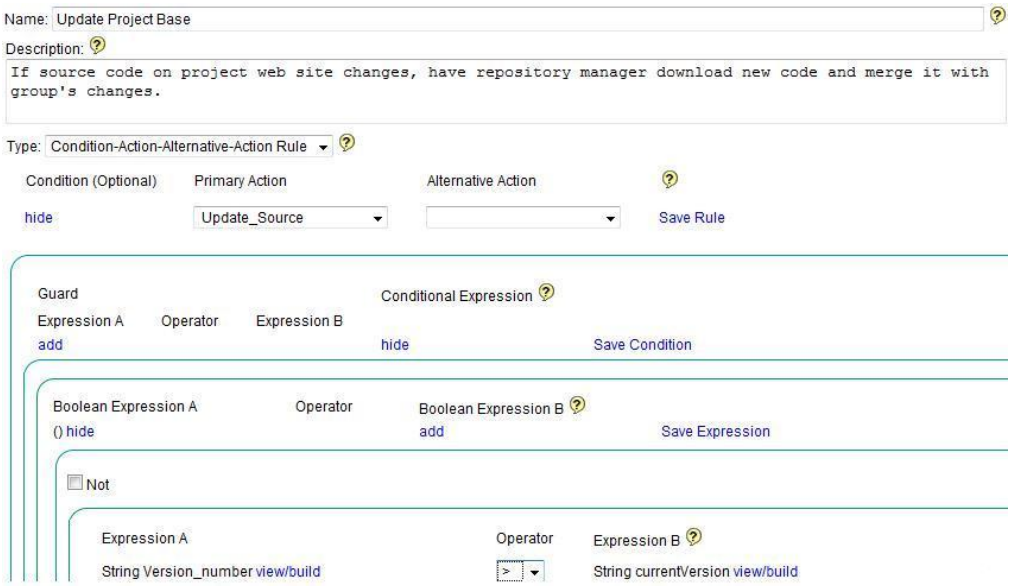

Figure 2. Rule definition interface

For operations, a user can either simply enter a URL for a web service that will be automatically invoked, or he/she can create a manual operation. For the latter option, the user must create a set of instructions for the user, a list of tasks to be completed, and a set of data attributes for the results that must be supplied. Also, one or more roles, and/or user names must be given so that the system knows to whom to send the notifications.

Both rules and operations can be combined with other rules/operations to form complex structures. An intuitive graphical interface allows the designer to create complex structures that dictate how the rules/operations are to be evaluated. These structures constitute the workflows that outline the main course of action within a project.

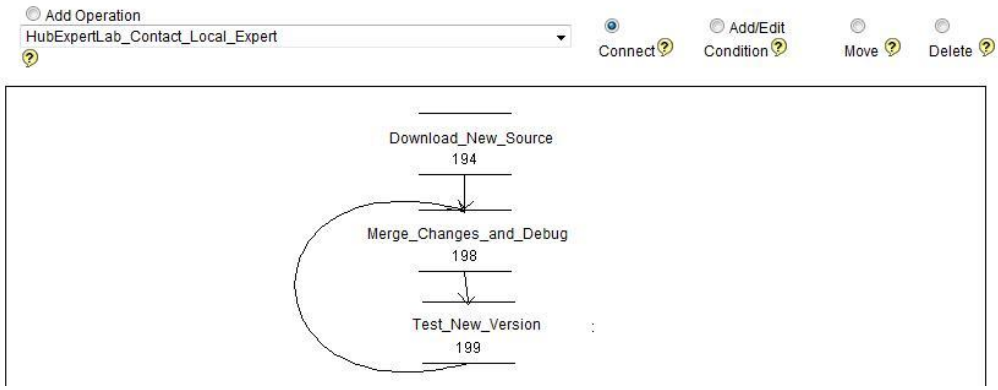

Figure 3. Operation structure definition interface

Though the interface for the definition of the project's structure is fairly involved, the end-user interface, which is all that the majority of students will ever see of the system, is very simple. The user has up to two options: 1. if the user belongs to a role which is capable of firing manual events, then the user will be able to select one of those

events and specify the required data, and 2. if the user belongs to a role which is responsible for completing an assigned operation, then there will be an interface available for signaling the completion of that operation and providing the necessary result data. The user will be alerted to the fact that he/she has been assigned an operation by an email, SMS, or notification by installed software. It will also be possible for such a user to view the current state of the workflows that incorporate any assigned operations.

6. Mobile Technologies

While students' internet access and usage tends to vary and it might be some time before a student receives an email notification that he/she needs to complete a given operation, mobile phones have the potential to provide nearly universal instantaneous access to information. A recent study found that 99.7% of American university students have some sort of device for mobile communications, and texting has overtaken email and instant messaging as the main way that such students communicate, with 94% using the technology; beyond that, 27% have smart phones that are capable of carrying out many of the same functions traditionally performed by an internet-ready PC (Ransford, 2009). With these statistics in mind, it makes sense to leverage the ubiquity of mobile phones to alert students to the tasks that are required of them, and allow them to respond to the system with the results of the tasks they perform.

Figure 4. SMS message to notify student of assigned operation

Because of its widespread availability, even in places where internet is not commonplace, text messaging would typically be the primary tool for communications between the system and the mobile devices of students involved in a project. When an operation is assigned, a message, or a sequence of several messages, with instructions and input data can be sent to all students who are assigned to a given role. This message can also include instructions for agreeing to perform the operation and for submitting results to the system; both of these actions will most likely involve sending a text message back

to the system with a code indicating the contents of the message. Assigning and responding to operations using SMS technology limits the possible input and output data to small bits of text or potentially small pictures (M. Shirali-Shahreza & S. Shirali-Shahreza, 2009), but many scenarios can be envisioned where this is all that is required. For more sophisticated devices, it is possible to install software that will allow the system to "push" alerts to the device. These alerts can be more complex, with input data consisting of full documents, pictures, and other media; the software can also present a variety of input fields to the user, and thus receive many different kinds of result data. In the case of our software engineering project, many of the tasks will require the delivery of code and other files, and operations for the project will likely involve a more sophisticated interface.

Support for mobile technologies will be particularly critical for introducing the described system to developing countries, where the availability of mobile phones and cellular networks is typically much more widespread than access to computers (Brown, 2003). Text messaging applications have already been used in such areas to help with language learning and health education and have tremendous potential for augmenting the existing education systems (Fotouhi-Ghazvini, Earnshaw, & Haji-Esmaeili, 2009).

Another advantage of incorporating the students' mobile phones is the additional data many of these devices are capable of collecting. Frequently, there is a built-in camera that can provide image data as part of the output data expected from a student's completion of an operation; Cavus and Uzunboylu touch upon one application where students capture pictures of objects in nature as part of an environmental education project (2008). It is frequently possible to capture sound as well, and, particularly in language-related activities, recordings can serve as useful output data. Many phones also come equipped with location detection capabilities, and geographic coordinates may be incorporated into an operation's results (Tan et al., 2009); this type of data would be especially pertinent in 'scavenger hunt' tasks, where a student or group of students is required to go to a certain location in search of something. If the system has access to the location data of the students involved, it can use this information to decide how to distribute operation assignments; if a student is closest to a given point, then it may be likely that he/she will be able to complete a given operation faster than students that are further from that point. The proximity of multiple students to each other might in itself be a sufficient criterion to fire an event and evaluate one or more rules (Martin, 2006).

7. Summary and Conclusion

Group assignments traditionally tend to be carried out using very ad-hoc processes. By looking to the fields of rule-based systems and workflow learning, a much greater degree of structure can be lent to students' activities. Rules and independent conditions within processes allow for a workflow to adapt to different situations, and incorporate various paths of execution depending on circumstances. This makes it possible to provide a learning experience that is both flexible and controlled.

This paper addresses how the ETKnet system can be applied to the collaborative learning domain, and uses a specific example to illustrate how a team project can benefit by employing a distributed event-and-rule-based system. The system combines event-triggered rules with workflow management to orchestrate the completion and delivery of a project, and to track the individual members, as well as the team as a whole. It promotes fast communication within and between teams, and has the potential to leverage mobile technologies to provide location-specific data and minimize any delays that may occur.

A powerful and intuitive user interface allows an instructor, or the students themselves, to specify the necessary rules and workflows with minimal training. The translation of rules and workflows into web services makes it possible for them to be shared across teams and be interoperable with rules and workflows defined by other teams. The replication of event and rule servers across multiple sites allows the system to scale up to support even the largest of classes. The system has not yet been tested in a classroom environment, but it is anticipated that there will be an opportunity to deploy the system in a software engineering class in the near future. If applied correctly, ETKnet has the ability to eliminate many of the deficiencies that tend to crop up in team projects, and to introduce new challenges that will force students to take a more active role in their projects' execution.

Acknowledgements

This work is supported by NSF under Grant IIS-0534065.

References

1. Brown, T. B. (2003). The role of m-learning in the future of e-learning in Africa? Paper presented at the 21st *ICDE World Conference*, Hong Kong.

2. Carro, R. M., Ortigosa, A., & Schlichter, J. (2003). A rule-based formalism for describing collaborative adaptive courses. *Knowledge-based Intelligent Information and Engineering Systems, 2774*(2)*,* 252–259.

3. Cavus, N., & Uzunboylu, H. (2008). A Collaborative Mobile Learning Environmental Education System for Students. *Proceedings of the 2008 International Conferences on Computational Intelligence for Modeling, Control and Automation; Intelligent Agents, Web Technologies and Internet Commerce; and Innovation in Software Engineering*, Vienna, Austria. 1041–1046.

4. Cesarini, M., Monga, M., & Tedesco, R. (2004). Carrying on the e-learning process with a workflow management engine. *Proceedings of the 2004 ACM Symposium on Applied Computing,* Nicosia, Cyprus. 940–945.

5. Degwekar, S., DePree, J., Beck, H. W., Thomas, C. S., & Su, Stanley Y. W. (2007). Event-triggered Data and Knowledge Sharing among Collaborating Government Organizations. *Proceedings of the 8th Annual International Conference on Digital Government Research*, Philadelphia, PA. 102–111.

6. Degwekar, S., DePree, J., Su, S. Y. W., & Beck, H. W. (2007). A Distributed Event-triggered Knowledge Sharing System: System Demo. *Proceedings of the 8th Annual International Conference on Digital Government Research*, Philadelphia, PA.

7. Dillenbourg, P. (1999). What do you mean by collaborative learning? *Collaborative-learning: Cognitive and Computational Approaches*, Elsevier, Oxford. 1–19.

8. Dobbins, P. (2007). *University of Florida Software Engineering 2007 Discussion Announcements*. Retrieved March 2, 2009, from http://www.cise.ufl.edu/class/cen3031su07/discussion/.

9. Fotouhi-Ghazvini, F., Earnshaw, R.A., & Haji-Esmaeili, L. (2009). Mobile Assisted Language Learning in a Developing Country Context. *In the Proceedings of the 2009 International Conference on CyberWorlds*, University of Bradford, UK. 391–397.

10. Gillies, R. M. (2007). *Cooperative learning: Integrating theory and practice,* Los Angeles, Sage Publications, 1–2.

11. Hayes-Roth, F. (1985). Rule-based systems. *Commun. ACM, 28*(9), 921–932.

12. Lin, J., Ho, C., Orlowska, M. E., & Sadiq, W. (2002). Using workflow technology to manage flexible e-learning services. *Educational Technology and Society*, *5*(4), 1–10.

13. Martel, C., Vignollet, L., Ferraris, C., David, J., & Lejeune, A. (2006). Modeling Collaborative Learning Activities on e-Learning Platforms. *Proceedings of the Sixth IEEE international Conference on Advanced Learning Technologies*, Kerkrade, The Netherlands, 707–709.

14. Martin, E., Carro, R. M., & Rodriguez, P. (2006). A mechanism to support context-based adaptation in m-learning. *EC-TEL* 2006, 4227, 302–315.

15. Odeh, S., & Ketanah, E. (2007). Collaborative Working e-Learning Environments Supported by Rule-Based e-Tutor. *International Journal of Online Engineering*, *3*(4), 20–26.

16. Ransford, M. (2009). Survey finds smart phones transforming mobile lifestyles of college students. *Ball State University Newscenter*. Retrieved from http://www.bsu.edu/news/article/0,1370,61565--,00.html.

17. Sahingoz, O. K., & Erdogan, N. (2003). RUBDES: A Rule Based Distributed Event System. *Lecture Notes in Computer Science*, *2869,* 284–291.

18. Shirali-Shahreza, M., Shirali-Shahreza, S. (2009). Sending pictures by SMS. *In the proceedings of the 11th International Conference on Advanced Communication Technology*, Phoenix Park, Republic of Korea. 222–223.

19. Stahl, G., Koschmann, T., & Suthers, D. (2006). Computer-supported collaborative learning: An historical perspective. In R. K. Sawyer (Ed.), *Cambridge handbook of the learning sciences* (pp. 409–426). Cambridge, UK: Cambridge University Press.

20. Tan, Q., Kinshuk, Kuo, Y. H., Jeng, Y. L., Wu, P. H., Huang, Y. M., Liu, T., Chang, M. (2009). Location-Based Adaptive Mobile Learning Research Framework and Topics. *In proceedings of 2009 International Conference on Computational Science and Engineering,* Vancouver, Canada. 140–147

21. Tsai, C. J., & Tseng, S. S. (2002). Building a CAL expert system based upon two-phase knowledge acquisition. *Expert Systems with Applications, 22*, 235–248.

22. Xiao, X., DePree, J., Degwekar, S., Su, S. Y. W., & Beck, H. W. (2008). Integrated Specification and Processing of Knowledge and Process for Achieving Knowledge Sharing among Collaborating Organizations. *Second International Conference on Information Systems, Technology and Management*, Dubai.