Web-service-based, Dynamic and Collaborative E-learning

Stanley Y. W. Su Gilliean Lee

Database Systems R&D Center
Department of Computer & Information Science & Engineering
University of Florida
Gainesville, FL. 32611 - USA
{su, glee}@cise.ufl.edu

Abstract

This paper describes an on-going effort to investigate problems and approaches for achieving Web-service-based, dynamic and collaborative e-learning. In this work, a Learning Content Definition Model is used to model distributed and sharable learning resources as *content objects*. Distributed and sharable software systems/components for supporting e-learning are modeled as *software objects*. Both types of objects are uniformly published as Web-services in a constraint-based Web-service registry and made sharable and reusable. An extended Web-service infrastructure provides a standard framework for the modeling, registration, discovery, binding and invocation of these objects. In this work, we also introduce a Learning Process Definition Model and a Learning Process Execution Engine for specifying and executing learning process models, which represent instructional modules in the forms of *activity trees*. An Event-Trigger-Rule Server is integrated with the Learning Process Execution Engine to make learning process models active, flexible, customizable and adaptable. It is also used to facilitate the interaction and coordination among learners, administrators, authors, and other personnel involved in collaborative e-learning.

1. Introduction

In recent years, there have been a number of initiatives in developing technologies for supporting Web-based learning. The Advanced Distributed Learning Initiative [1], the IMS Global Learning Consortium [2], the Schools Interoperability Framework [3], the Open Knowledge Initiative [4], MIT's OpenCourseWare [5], and Dspace being jointly developed by MIT and the Hewlett-Packard Company ([6], [7]) are a few examples. The Sharable Content Object Reference Model (SCORM) ([8], [9]) is a reference model initiated by the Advanced Distributed Learning (ADL) program of the Department of Defense (DoD) and the White House Office of Science and Technology Policy (OSTP) in November 1997. Its specification is composed of a Content Aggregation Model for aggregating learning resources to form leaning modules and courses, and a Runtime Environment for launching learning resources and enabling the communication between learning resources and Learning Management Systems (LMSs). According to the specification, it is envisaged that Internet users and heterogeneous LMSs would use the Web as a universal platform for accessing and launching sharable content objects and for establishing close communication, interaction and coordination among content object developers, course authors, content users, and course administrators. To realize this vision, sharable content objects must be durable, interoperable, accessible and reusable. These requirements impose the following requirements on the network of heterogeneous web-based LMSs [8].

- The ability of a web-based LMS to launch content that is authored by using tools from different vendors and to exchange data with that content.

PGLDB'2003, pp. 68-77, 2003.

- The ability of web-based LMS products from different vendors to launch the same content and exchange data with that content during execution.
- The ability of multiple Web-based LMS products/environments to a access common repository of executable content and to launch such content.

In order to meet the above requirements, it will be necessary to have a uniform way of modeling, not only learning resources, but also heterogeneous learning tools and LMSs as well as an information infrastructure to enable the interoperation and sharing of their contents and functionalities. Also, the aggregation model that defines the learning sequence or process has to be flexible, adaptable and customizable to meet different learners' needs and learning contexts. Furthermore, there must be a general and powerful mechanism for facilitating the close interaction, communication and coordination among learners, content authors, course administrators, assistants, etc., to achieve ***collaborative e-learning.*** Research and development work to meet the above needs is consistent with the vision and goals of the Advanced Distributed learning (ADL) program.

In this work, we present some problems and approaches to integrate the Web-service technology, a dynamic learning process modeling and management technology, and an event-trigger-rule technology for achieving Web-service-based, dynamic and collaborative e-leaning. The focus of this research is on 1) the development of techniques for the modeling, registration, discovery, binding and invocation of distributed learning objects, 2) the development of a Learning Content Definition Model (LCDM) for modeling content objects, a Learning Process Definition Model (LPDM) for modeling learning processes, and a Learning Process Execution Engine (LPEE), which is paired with an Event-Trigger-Rule (ETR) Server, for making learning process models active, adaptable, flexible and customizable, and 3) the integration of event, event filtering, event notification, condition-action rule processing technologies with computer mediated communication (CMC) tools to enhance the interaction, coordination, and communication among people and software systems that are involved in distributed e-learning.

2. Problems and Approaches

We describe the three focused R&D efforts below:

2.1 Modeling, registration, discovery, binding and activation of sharable learning objects

In order for the e-learning community to share multimedia learning resources as well as heterogeneous learning tools and components of software systems (e.g., LMSs), there must be a standard and uniform way for the developers of these resources, tools and software systems to model them and publicize their availability for sharing in the e-learning community. A very general and powerful way, which is adapted from the object-oriented technology, is to uniformly model them as "***learning objects***" having attributes/properties (i.e., meta-data) and operations/methods (i.e., behaviors). The learning objects that model content resources with behaviors are then called "**content objects**" in this paper, and those that model software tools and systems/components are called "**software objects**". In our Learning Content Definition Model (LCDM), we model each content object in terms of pre-assessment items, content items, practice items, post-assessment items, and constraints. The pre-assessment items (optional) are

used to assess a learner's prior knowledge with respect to the content items to be presented and studied by learners. The practice items (optional) contain items to be exercised by learners after they have studied the content items. The post-assessment items are used to test the proficiency of the learners on the contents of the content items. The constraints are used to specify such things as users' profile, learning context, etc., in which the content object is suitable for use. The behavioral specification of a content object provides the Get functions for accessing the four types of items separately as well as for accessing the entire content object as a whole. These functions allow an LMS to import either a part of or the entire content object into its processing environment. We note here that the last three types of items form the Reusable Information Object of Cisco's Reusable Learning Object Strategy [10]. Software objects model those functionalities of some existing application systems, LMSs, or other software systems that users/organizations are willing to share with others. The specifications of these objects provide the information to enable either remote or local invocations of these functionalities programmatically. All learning objects (content and software objects) are executable in that their operation specifications can be mapped to their corresponding APIs of their local, heterogeneous implementations.

In addition to modeling learning objects in terms of attributes and operations, we believe that it is important also to specify the constraints associated with their attributes and operations. The constraint specification of an object can be constraints on the valid values of its attributes, on the inter-attribute relationships, or on the values of the parameters associated with its operations. For example, the constraint specification of a content object may include such information as its prerequisites, difficulty level, cost, the estimated learning time, and other value constraints on its descriptive attributes. The meta-data and behavior information and their associated constraints can then be registered with and maintained by online registries (or repositories) and be searched either manually by content users or programmatically by learning tools and LMSs for the discovery of the published objects either for the downloading purpose or for remote executions. The constraint associated with the meta-data and behaviors would allow more accurate selections of sharable learning objects that satisfy content users' or software requestors' requirements. As pointed out in [8], the standardized way to register and discover content objects will enhance opportunities for their reuse. The constraint specification and processing, which is not a part of the traditional object model or object-oriented systems, would enhance the discovery of suitable learning objects to meet different users' needs. Another important SCORM's requirement is for content objects to be interoperable across multiple LMSs and the run-time environment should facilitate such interoperations [9]. To meet this requirement, we need also a standard protocol to bind a request to a discovered learning object and to launch or activate its operations at run-time.

To meet the modeling, registration, discovery, binding and invocation requirements discussed above, we believe that the Web-service technology being developed by the IT industry can be adapted and extended to serve this purpose. Universal Description, Discovery and Integration [11] provides the general framework to allow both content objects and software objects to be defined as Web-services using the Web Service Description Language [12]. WSDL is XML-based and can be extended to include the specification of meta-data, behaviors and their constraints. A WSDL document in XML format can be used to define any type of learning objects and be posted on the Internet. Its access point (URL) together with other meta-information can be registered with an online Web-service registry. The contents of the registry can then be searched manually or programmatically to discover and select suitable learning objects. As an alternative to

downloading some identified objects for local use, the Web-service model allows the run-time binding of a service request to a remote Web-service and provides a Simple Object Access Protocol [13] to activate the remote service. By applying the Web-service technology, learning objects become Web-services, which model different types of learning resources, learning tools, learning management systems/components developed by different authors and vendors. They can interoperate and be activated locally or remotely based on their published interfaces. Since the contents of the registry can evolve over time (i.e., some registered objects may have been modified or deleted, and new objects may have been registered), dynamic binding and remote access and activation of learning objects can enhance their delivery and use.

The existing specification and implementation of UDDI does not allow the specification of constraints associated with Web-services. Without this specification capability, a learning object can be selected by the registry for downloading or for online use, but does not satisfy a requestor's requirements. In this work, we have extended the WSDL and UDDI's capabilities by allowing the specification of constraints associated with learning objects. Requestors' requirements can also be specified as constraints in a constraint specification language. We use a Constraint Satisfaction Processor developed at the Database Systems R&D Center to store constraints and perform constraint matching to select suitable Web-services [14]. This processor was also used as a component in an automated negotiation system developed for e-business applications [15]. The processor does not replace the current UDDI implementations but complements their capabilities. It can be integrated with an existing implementation of UDDI (e.g., IBM's UDDI v2.0) to enhance its functionalities.

2.2 Modeling and management of dynamic learning processes

The process of creating and delivering learning experiences involves the creation, discovery, and aggregation of simple electronic assets into more complex learning resources and then organizing those resources into a predefined sequence of delivery [16]. The Sequencing Definition Model described in [17] allows a learning activity to be defined as an ***activity tree*** (a hierarchical organization of learning content) having sequencing control modes associated with its items called ***activities.*** Each activity can have a number of condition-action-type of sequencing and roll-up rules, which control the behavior of an LMS, based on the data that keeps track of the statuses on the progress toward reaching the objective of learning contents and the progress of learners' activity/attempts. The tree structure, the control modes and the sequencing rules constitute a sequencing model, which specifies the structure and the conditions (or rules) in which content objects should be launched in certain sequences and control modes.

A learning module or course defined by an activity tree is static in the sense that it has a fixed structure, predefined control modes, and sequencing and roll-up rules, which are to be followed by an Instruction Management System (IMS) for each and every enactment of the tree. (i.e., all instances of its processing). This is not ideal because different content users may take the module or course with different background training, in different learning contexts and constraints, and for different needs (e.g., different background trainings require different difficulty levels or different orders of content delivery, or some component may be skipped due to a learner's previous knowledge). It is unreasonable to expect that an activity tree can be predefined by a content developer to suit the different needs and constraints of all potential learners. An activity tree can only represent a "typical" structure of learning for a group of potential

learners. It has to be customized to meet individuals' needs, constraints and learning contexts. We believe that a learning management system controls the processing of activity trees (i.e., the sequencing models) must be **dynamic** in the sense that they are **active**, **flexible**, **adaptive** and **customizable**.

To achieve the above four dynamic properties, we adopt and extend our prior work on a dynamic workflow management technology reported in ([18], [19], [20]). We introduce a Learning Process Definition Model (LPDM) for modeling learning modules and sub-modules of different granularities as activity trees; the same as SCORM's Content Aggregation Model [16] but with a number of extensions to make these process models active, flexible, adaptive and customizable. In our work, an activity tree defines a learning process model. Each activity specification contains an activity name, an activity identification, a textual description to explain the meaning of the activity, a content object and its binding information, and a set of optional condition-action rules. These rules may contain 1) pre-activity rules to be processed before the enactment of the activity, 2) pre-assessed rules to be processed after the pre-assessment task and before the content object is accessed and presented to the learner, 3) drill-down rules to be processed after the practice items of the content object has been presented to the learner and before leaving the activity to go down the activity tree, 4) roll-up rules to be processed when the activity is revisited during the roll-up process and before the roll-up content items and post-assessment items are presented to the learner, and 5) post-assessed rules to be processed after the post assessment task to evaluate the result of post-assessment. Since the condition parts of these rules may check the user profiles and the progress they made at the stage of processing the activity, only some of these rules are applicable to a specific learner or a category of learners. The author who defines the activity can customize the rules to suit different learners or categories of learners. The activity specification described above is compiled into the following program code for processing at run-time. The run-time code performs a sequence of "drill-down" tasks and a sequence of "roll-up" tasks. The drill-down tasks are: 1) post a pre-activity event, which would trigger pre-activity rules, 2) access and present the pre-assessment items of the associated content object and perform assessment, 3) post a pre-assessed event to signal the completion of the pre-assessment, which triggers the processing of pre-assessed rules, 4) access and present the content items of the content object (e.g., an overview for a non-leaf activity or learning content for a leaf-activity), 5) access and present practice items to the learner (for leaf activity only), and 6) post a drill-down event, which triggers drill-down rules. The roll-up tasks are: 1) post a roll-up event, which triggers roll-up rules, 2) access and present roll-up content items (e.g., a summary of content items presented in the sub-tree of a non-leaf activity), 3) access and present post-assessment items and perform assessment, and 4) post a post-assessed event, which triggers post-assessed rules. We note here that the drill-down tasks 1,3 and 6 and the roll-up tasks 1 and 4 post events to signal the stages of a learning process in each activity. These events are automatically generated. They replace the rules specified in the activity specification. Condition-action rules triggered by these events are stored and managed by an **Event-Trigger-Rule (ETR) Server** in our implementation. This separation allows drill-down and roll-up rules to be modified and/or added without affecting the activity code; an approach that is critical for achieving the dynamic properties of learning process models, to be addressed later.

Learning process models are processed by a Learning Process Execution Engine (LPEE). An enactment of a learning process model by a learner forms a learning process instance (or an instance of the activity tree). Multiple instances of the model

can be concurrently processed by LPEE to control the learning activities of multiple learners, who are taking the same learning course/module.

The key extensions we made to the SCORM's Content Aggregation Model are as follows. First, the modeling construct called Activity is extended to include a Web-service request made either to a specific content object or to a Web-service registry for a dynamical binding of the request to a registered content object or to a software object that returns a content object. Web-services requests can also be issued in the action part of a condition-action rule. Some simple computational statements such as assignments and comparisons can also be used in the action part of the drill-down and roll-up rules to store and test the data returned by the executable learning objects. Second, the sequencing control modes proposed in [17] specify the choosing and sequencing of child activities under a parent activity (i.e., choice, flow, forward only, choice exit, or a combination of these modes). They are treated as a part of activity specification. In our LPDM, they are specified in a different modeling construct called "Connector" for the following three reasons. One, semantically, they actually specify the "relationships" between the parent and child activities or among child activities instead of the activity itself. Two, by specifying the sequencing control modes in a separate modeling construct, modifications made to the activity or the connector will not affect the other. This will enhance the reusability of activity and connector specifications. The third reason, a more important one, is that the separation will allow the control structure of an activity tree to be modified at run-time for each enactment of the tree in order to fit each learner's background, needs, constraints, and learning context. This is essential for making a learning process model customizable, an important property to be addressed in the next paragraph. The third extension to SCORM is to allow non-leaf nodes of an activity tree to contain content items. This is useful for providing an abstract and/or an introduction to learners before the activity tree rooted by a non-leaf activity is launched, and a summary of the content items of the activity tree after the tree is processed. It also allows all the activities to be modeled uniformly by the items and rules given in the preceding paragraph. Fourth, SCORM's sequencing model allows ***sequencing and roll-up rules*** to be specified as a part of activity specification. These rules are in the form of condition-action rules, and are evaluated at different times in the sequencing and content delivery process. They are defined based on a set of predefined keywords such as Satisfied, Attempted, Skip, Retry, etc. The action part of a sequencing rule can specify a precondition action, and a post-condition action or an exit action, which are time-wise relative to the learning activity specified by an activity. In our work, we use a general rule specification language to specify what we call drill-down and roll-up rules. The language can specify not only SCORM's sequencing and roll-up rules but also rules for handling exception conditions and/or enforcing security and integrity constraints, policies and regulations of the learning module/course's authors and/or administrators. As we have mentioned before, different types of rules are stored and managed by an ***Event-Trigger-Rule (ETR) Sever***. They are triggered for processing when events are posted during the execution of an instance of an activity tree. This approach allows drill-down, roll-up and other types of rules to be modified and new rules to be added and triggered by the occurrences of events, without entailing changes to the generated activity code, based on activity specifications. Event-injected activities, condition-action rules and triggers, which link events to rules, can be separately specified by different people or organizations and be managed in a distributed fashion (see a further discussion in Section 2.3). Changes made to one type of specification will not affect the other. This separation is important for achieving the dynamic properties at run time to be elaborated upon next.

In addition to the model extensions described above, our approach to achieve the dynamic properties at run time is to integrate the ***ETR Sever*** with the ***Learning Process Execution Engine***. When a learning process model represented by an activity tree is enacted by the process execution engine, an instance of the model is established. The execution of an activity would post events before processing the activity, after the pre-assessment, before the drill-down process, before the roll-up process, and after the post-assessment of the activity to trigger the execution of rules. These rules can, not only affect different sequencing behaviors and/or enforce different constraints, policies, and regulations and/or handle exception situations, but also conditionally modify the process model instance itself at run time to change the control structure of content delivery. This run-time dynamism can be achieved by the following implementation technique. Since we separate the specifications of activities from those of Connectors, which now contain the specifications of sequencing control modes, we can compile an activity tree into two part: ***activity code*** for activities, which post events to trigger rules and launch the instructional activities specified in activity specifications (including Web-service requests for local or remote learning objects) and the ***control structure*** of the activity tree and its sequencing control modes. The Learning Process Execution Engine is driven by the contents of the control structure. Rules that are defined for a specific learner or a category of learners can be conditionally activated at run-time to modify the copy of the control structure created by an enactment of an activity tree (e.g., a leaner initiates an attempt of a learning module). Thus, the original activity tree can then be customized to suit each individual learner's profile and learning context. Rules managed and processed by the ETR Server can also be modified at run-time. A modified rule can be compiled into Java code and the code can be automatically loaded for execution using Java's class loading facility.

The **learning process models,** which are processed by the Learning Process Execution Engine and the ETR server, are **active** because the enactment of a process model (i.e., an activity tree) can automatically activate rules to perform any desired operations at different stages of a learning process. They are **flexible** because the Web-service requests for learning objects included in activity specifications can be dynamically bound to the operations of local and remote learning objects. They are **adaptable** because each instance of a process model can trigger rules to modify the control structure of that instance to fit a particular content user's profile, needs and learning context (e.g., by conditionally enacting a supplemental learning module not included in the original model, skipping some activities, changing the request for a content object to one that has a lower or higher difficulty level, altering the condition or action of a rule, etc.). They are also **customizable** because different sets of rules defined for different users or categories of users can be triggered by the same set of events posted by the same activity tree in different enactments of the process model (i.e., different model instances).

The research work and approaches described above are based on our previous research on event-trigger-rule processing and dynamic workflow management ([21], [18], [19], [20]) as well as the concept, model and language introduced for workflow management by the Workflow Management Coalition [22]. We envision that the ETR Server and the Learning Process Execution Engine are critical components of a web-based learning management system. They, together with the learning content definition model and the learning process definition model presented above, will make the learning management system dynamic, having the above four dynamic properties.

2.3 Facilitating collaborative learning

Internet-based learning technologies emphasize interactions and collaborations ***among people*** because learning is a largely a social activity; even the most well developed multimedia interactive materials lack the flexibility of human interaction [23]. Collaborative learning requires that each learner taking a leaning module/course can do group work with fellow learners and is able to establish close communication and coordination with lecturers, assistants, mentors, and administrators. Computer mediated communication (CMC) tools such as email, mailing list, computer conferencing, video conferencing, Usenet news group, Internet relay chat, etc., can be used to enhance communication and coordination. However, the use of these tools needs to be integrated into the learning process. That is, during the progress of taking a learning module or course, and at the proper times and under proper conditions, people involved in collaborative learning need to be advised to make use of CMC tools to establish the communication with one another. The results of using these tools should be fed back to the learning management system (LMS) that controls and coordinates the learning process. A mechanism is needed to tie the CMC tools with LMSs.

In our work, we use the same event-trigger-rule mechanism used to achieve the dynamic properties of learning process models in order to achieve this purpose. Using the graphical user interface provided by the ETR Server, we can define anything of interest in e-learning as an event, and the condition to be verified and the action to be taken upon the occurrence of the event as a rule. The relationship between the event and the rule is specified by a trigger. For example, a content user who is taking a course encounters a specific problem (an event), which would trigger a rule to post a message to a discussion board for assistance from some other people. Another example, a potential content user, who searches the registry and fails to find a desirable content object, would subscribe to the New-Object event or the Modify-Object event, and specify an event filter to indicate the condition(s) under which he/she would like to be notified by email or other means. Thus, when a new content object is registered with the registry (i.e., the occurrence of the New-Object event) or an existing content object is modified (i.e., the occurrence of the Modify-Object event), if the event filter condition(s) is satisfied, the user will automatically be notified. Many content users can subscribe to the same event with the same or different event filters. When the event occurs, they can all be notified if their filtering conditions are satisfied. A third example, when several students of an online course are working on a team project and one of the team members encounters a problem (an event), a rule would be triggered to send an email message to all team members to advise them of the time to participate in a video conferencing session to discuss and resolve the problem.

In this work, we have developed servlets that are accessible through Web browsers to allow people involved in e-learning to define and register events in a Web-service registry (or registries) just like the registration of learning objects discussed in Section 2.1. These people can also browse or programmatically search the registered events and subscribe to them. Rules and triggers can be defined by them and stored at their corresponding sites with security and privacy access control. They will be processed by the replicas of the Event-Trigger-Rule Server that are installed at these sites. The processing of distributed triggers and rules is activated by the event notification mechanism. Event, event filtering and event notification are powerful means to tie loosely coupled systems and geographically distributed systems and people together. By integrating them with the rule specification and processing mechanisms, learning process modeling and execution mechanisms, and CMC tools, they can together form a

powerful information infrastructure for achieving advanced distributed learning. In this research, we are investigating the best way to integrate these tools and mechanisms. This effort is based on our expertise and experiences in building event-trigger-rule-based systems ([24], [25], [26], [27]).

References:

[1] Advanced Distributed Learning Initiative, http://www.adlnet.org

[2] IMS Global Learning Consortium Inc., http://www.imsglobal.org

[3] Schools Interoperability Framework, http://www.sifinfo.org

[4] O.K.I. and Eduworks, OKI White Paper; What is the Open Knowledge Initiative? http://web.mit.edu/oki/product/whtpapers/whatis.html

[5] http://ocw.mit.edu/global/about-ocw.html

[6] http://dspace.org/what/definition.html

[7] http://chronicle.com/free/v49/i15/15a03101.htm

[8] Advanced Distributed Learning, Sharable Content Object Reference Model Version 1.2: The SCORM Overview, October 1, 2001

[9] Advanced Distributed Learning, Sharable Content Object Reference Model Version 1.2: The SCORM Run-Time Environment, October 1, 2001

[10] Barritt, C., "Reusable Learning Object Strategy version 4.0",Cisco Systems, Inc., Nov. 2001.

[11] Universal Description, Discovery, and Integration (UDDI), http://www.uddi.org/, http://uddi.microsoft.com/ and http://www-3.ibm.com/services/uddi/

[12] World Wide Web Consortium (W3C), "Web Services Description Language (WSDL) 1.1", http://www.w3.org/TR/wsdl.

[13] World Wide Web Consortium (W3C), "Simple Object Access Protocol", http://www.w3.org/TR/SOAP/, and http://www.develop.com/soap/

[14] Degwekar, S., Lam, H. and Su, S. Y.W., "Constraint-based Brokering for Publishing and Discovery of Web-services," Technical Report, Database Systems R&D Center, University of Florida, 2002.

[15] Su, Stanley Y. W., Huang, C., Hammer, J., Huang, Y., Li, H., Wang, L., Liu, Y., Pluempitiwiriyawej, C., Lee, M., and Lam, H., "An Internet-based Negotiation Server for E-Commerce," *VLDB Journal,* Vol. 10, No. 1, Aug. 2001, pp. 72-90.

[16] Advanced Distributed Learning, Sharable Content Object Reference Model Version 1.2: The SCORM Content Aggregation Model, October 1, 2001

[17] SCORM Version 1.3 Application Profile – Draft

[18] J. Meng, S. Y. W. Su, H. Lam, and A. Helal, "Achieving Dynamic Inter-Organizational Workflow Management by Integrating Business Processes, Events, and Rules," Proceedings of the 35th Hawaii International Conference on System Sciences, Hawaii, USA, January 2002.

[19] Meng, Jie, Krithivasan, Raja, Su, Stanley Y.W., and Helal, Abdelsalam, "Flexible Inter-enterpriseWorkflow Management using E-Services," Proceedings of the 4th IEEE International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems, June 2002, California, USA.

[20] Su, Stanley Y. W., Meng, J., Kristhivasan, R., Degwekar, S. and Helal, S., "Dynamic Inter-Enterprise Workflow Management in a Constraint-based E-service Infrastructure," Electronic Commerce Research, 3:9-24, 2003.

[21] Su, Stanley Y. W., Lam, H., Lee, M., Bai, S., and Shen, Z., "An Information Infrastructure and E-services for Supporting Internet-based Scalable E-business Enterprises," Proceedings of the 5th International Enterprise Distributed Object Conference (EDOC 2001), Seattle, WA, Sept. 4-7, 2001, pp.2-13.

[22] Workflow Management Coalition, "Interface1: Process Definition Interchange V 1.1 Final (WfMC-TC-1016-P)," 1999; http:// www.wfmc.org.

[23] http://www.warwick.ac.uk/ETS/Publications/Guides/Printer.htm

[24] Lam, H. and Su, Stanley Y.W., "Component Interoperability in a Virtual Enterprise Using Events/Triggers/Rules," Proc. of *OOPSLA '98* Workshop on Objects, Components, and Virtual Enterprise, Vancouver, B.C., Canada, Oct. 18-22, 1998, pp. 47-53.

[25] Su, Stanley Y. W. and Lam, Herman, "IKnet: Scalable Infrastructure for Achieving Internet-based Knowledge Network," invited paper, Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, l'Aquila, Rome, Italy, July 31-Aug. 6, 2000.

[26] Lee, M., Su, S.Y.W., and Lam, H., "Event and Rule Services for Achieving a Web-based Knowledge Network," Proc. 2001 Int. Conference on Web Intelligence (WI-2001), Maebashi City, Japan, Oct. 23-26, 2001, pp. 205-216.

[27]Lee, Minsoo, Su, Stanley Y. W., Lam, Herman, "A Web-based Knowledge Network for Supporting Emerging Internet Applications," the WWW Journal, Vol. 4, No. 1/2, 2001, pp. 121-140.