ERF: An Event-Rule Framework for Supporting Heterogeneous Distributed Systems

TABLE OF CONTENTS

1. INTRODUCTION ........................................................... 1

2. JUSTIFICATION FOR THE PROPOSED RESEARCH ....................... 4

3. RELATED WORK ............................................................ 5
   3.1. HDS Support Environments and Tools .............................. 5
   3.2. Event Services ..................................................... 6
   3.3. Event-Rule Approaches .......................................... 7

4. DEVELOPMENT AND EXPERIMENTATION ............................... 8
   4.1. RUBIES: A Rule-based Intelligent Event Service .................. 8
       4.1.1. Overview ............................................... 8
       4.1.2. Distribution architecture .................................. 9
       4.1.3. Event model ............................................. 9
       4.1.4. Rule model and specification language ..................... 10
       4.1.5. Interface and behavior of RUBIES ........................ 11
       4.1.6. Research issues ....................................... 12
   4.2. An Integrated Development Environment ......................... 13
       4.2.1. Objectives ............................................. 13
       4.2.2. Proposed tools ........................................ 13
       4.2.3. Research issues ....................................... 14
   4.3. Validation and Demonstration: Real-time Flood Alert System (RTFAS) . 16
       4.3.1. Current system ....................................... 16
       4.3.2. An event-driven alert system ............................ 18
       4.3.3. Event and rule examples ................................ 19
       4.3.4. The user interface ..................................... 21
       4.3.5. System Validation ..................................... 21

5. RESEARCH INFRASTRUCTURE .............................................. 22

6. INTEGRATION WITH EDUCATION .......................................... 23

7. RESEARCH AND EDUCATION PLAN ..................................... 24
   7.1. Project schedule and milestones .................................. 24
   7.1. Management plan .............................................. 28

8. COST SHARING BY UPRM .................................................. 29

9. EXPECTED RESULTS AND IMPACT OF PROPOSED RESEARCH ............ 30

10. RESULTS FROM PRIOR NSF FUNDING .................................... 31
1. INTRODUCTION

Although a number of standards for supporting heterogeneous distributed systems (HDS) already exist (e.g., RPC, CORBA, DCE, DCOM, Java RMI), there is still a lack of abstractions, services and tools for specifying, designing, implementing, monitoring, debugging and maintaining a HDS. For an effective support of these activities, a conceptual view of a distributed system is needed. We mean by conceptual view the specification of the system as seen by the community of developers, in terms of structure and behavior. It should be noted that existing environments (or “middleware”) do well in specifying structure, i.e., attributes and structural relationships among system components. However, in terms of behavior, the specification is limited to the definition of function- or method- signatures. The semantics of behavior are hidden or “buried” inside the application code (implementation). Therefore, anyone interested in knowing the behavioral semantics of the system either has to look into the application code, or into a specification or design document (which probably will be “out of sync” from the implementation). Furthermore, existing environments do not allow to incorporate changes in behavior dynamically; any change in the behavior will involve changes in the implementation of functions or methods, which in many cases requires recompilation. On the other hand, the specification and implementation of a HDS - including structure and behavior- may be specified at a high level, maintained and executed in an environment which provides (i) high-level abstractions (rules) for specifying behavior in terms of events, conditions and actions; (ii) ability to incorporate changes dynamically; (iii) an engine for specifying and processing rules that react to events in real time; and (iv) tools for specifying, designing, implementing, debugging, monitoring and maintaining the system.

This proposal presents a plan for the development of an Event-Rule Framework (ERF), which comprises a set of services and tools for supporting the development of HDS. Our research aims to study design alternatives, algorithms, applications and implementation issues to accomplish the following objectives:

Object-oriented model. Similar to CORBA, DCOM and Java RMI, ERF should have an object-oriented model in which system components are treated as objects. ERF does an extension to existing models by treating events and rules as objects.

Support of multiple standards. ERF should be designed to support the multiple standards for distributed system environments (CORBA, Java RMI, DCE and DCOM). An open architecture will allow modifications for supporting new standards as well.

---

1 by permission from the program director to exceed the 15-page limit for this multi-PI proposal
**Events.** ERF will use events as an abstraction for specifying system behavior. The specification of behavior is done in terms of events that trigger rules. A formal object-oriented event model will allow the systematic definition of events.

**Rules.** In ERF, ECAA (Event-Condition-Action-AlternativeAction) rules are used to specify system behavior. Rules are defined in terms of trigger events, conditions that need to be satisfied to apply the rule, and a set of actions and alternative actions to perform when the events occur and the conditions are satisfied. A formal object-oriented rule model will allow the design and implementation of a rule support system independent of any particular rule language syntax.

**Intelligent event service.** The heart of ERF is a Rule-Based Intelligent Event Service (RUBIES). Following the object-oriented paradigm of existing standards, RUBIES comprises a set of classes whose interfaces include methods for rule definition, event notification (or posting), registration of event producers and consumers, and rule management.

**High-level specification model.** A high-level specification model, such as that of UML [Booch98], Zoom [Anlgano95] or HeNCE [Wolski95] (or extensions to them) is needed to formally define the structural and behavioral semantics of a HDS. This will allow the use of graphical tools for specification, implementation, debugging, monitoring and maintenance of the system. A formal model will allow the automated or semi-automated translation into an implementation.

**Specification tools.** ERF will provide an environment for specifying: (i) partitioning of a HDS into its components and their relationships; (ii) event and event-flows among system components; and (iii) rules and their distribution among distributed event services.

**Monitoring and debugging tools.** During run time in a real application environment, monitoring and debugging tools will be needed to monitor the event occurrences and event flows, the state of the system and the execution of rules. Such tools will assist in the debugging of the system.

The proposed research activities can be categorized into four highly-coupled components, namely (i) development and experimentation, (ii) validation and demonstration, (iii) research infrastructure, and (iv) integration with education.

**Development and experimentation.** We propose the development of a system which will accomplish the objectives mentioned above. The development of such system involves experimentation to study design alternatives, algorithms and implementation issues. The system will serve as a test bed for further research on HDS.

**Research infrastructure.** To ensure the success of this project, and with the intention to enhance the research infrastructure of the ECE at UPRM, we request funding for the creation of a Heterogeneous Distributed Systems Laboratory (HDSL).
**Validation and demonstration.** One important component of this research is its validation and demonstration by using the system in the development of a real-life application. We are proposing the use of ERF in the development of a Real-Time Flood Alert System (RTFAS). This will be an interagency software development project in which the US Geological Survey of P.R., the Aqueduct and Sewer Authority and the Civil Defense will collaborate with UPRM. An iterative collaboration process will help in the specification of requirements, design and implementation of the framework.

**Integration with education.** We propose a set of activities that will enhance education through (i) development of undergraduate and graduate research projects, (ii) enhancement of graduate curricula,(iii) supervision of graduate and undergraduate students, and (iv) dissemination of research.

Figure 1 presents a process flow diagram of the leading research and development activities. Notice that all the activities are highly coupled, which implies a strong collaboration among investigators. Each one of these activities will be described in more detail in this proposal.
The rest of this proposal is organized as follows. In the next section, we present a justification for the proposed research. Section 3 presents a survey of related works. In Section 4, we present our development and experimentation plan. The research infrastructure development plan is presented in Section 5, where we propose the creation of a Heterogeneous Distributed Systems Laboratory (HDSL) for supporting the proposed and future research. The plan for integration with education is presented in Section 6. In Section 7, we present a master plan and schedule. Costs to be shared by our institution (University of Puerto Rico - Mayaguez) are described in Section 8. The expected results and impact of the proposed research are presented in Section 9. Results from prior NSF funding are presented in Section 10.

2. JUSTIFICATION FOR THE PROPOSED RESEARCH

The proposed research is motivated by existing event services and the needs for a powerful event model, a common event handling specification and an integrated development environment.

Events and event services. Works on existing HDS standards (like CORBA and DCE) have recognized the importance of events as means of achieving higher levels of behavioral abstraction. Events (or autonomous asynchronous occurrences [Bacon95]) have been successfully used as a paradigm for the specification of behavior in graphical user interfaces, real-time systems and active databases. The behavior of a distributed system can be defined at a high level in terms of which events are interesting, under which circumstances they are relevant, and how system components react upon the occurrence of the events. An event service is a system component that has methods for the definition of relevant events, the registration of event producers and consumers, and the notification (or posting) of event occurrences from producers to consumers, or event requests from consumers to producers.

The need for a powerful event model. As the complexity of a HDS increases, the number of interactions among system components also increases. When events are used to define such interactions, a lot of highly interrelated events occur. Composite events can be used as abstractions for such interrelations. Examples of composite events are: (i) event e2 happens after event e1; (ii) events e1, e2 and e3 happen, regardless of their order; (iii) events e1 and e2 happened separated by a time interval t12, (iv) events e1 and e2 happened, but not event e3. Time specification of events is also needed. For example, an event might be relevant only at a given time (absolute time) or during a time period (relative time). Time is also needed for scheduling of event handling; in many applications, the occurrence of events is relevant only at specific times (or dates), during specific time (or date) intervals, or with a specific frequency (e.g., every 10 minutes). An event-rule framework should have the flexibility for supporting such facilities.

The need for a common event handling service. One common approach of existing systems is to do event handling at the object or “functional component” level. In this approach, each object that is registered as a consumer of an event is informed by the event service of the event occurrence, and reacts to the event by executing an event-handling function (or method). The functionality of existing systems is limited to the posting and forwarding of events to registered consumers which provide event handling functions, implemented in their own language. This
approach has many disadvantages. Firstly, the semantics of event handling are "buried" inside the service's application code and represented in the service's implementation language, instead of being specified at a higher level (e.g., system management) by means of a common language. Secondly, any change in behavior of event handling will require a change in the implementation of the event handler, which in most of the cases will require recompilation. Thirdly, it does not allow dynamic changes in the specification of behavior, but rather static changes which need to be implemented in the service's implementation language and recompiled. On the other hand, a common event handling specification language (e.g., a rule language) would allow sophisticated end users to specify, in a declarative manner: which events are relevant, under which circumstances they are relevant, which other services are needed to handle the event, and which sequence of actions to take upon the occurrence of the event. If rules are interpreted, changes would occur dynamically without the need to recompile any service. An intelligent event service may provide such facilities.

The need for an integrated development environment (IDE). There is a lack of abstractions, services and tools for specifying, implementing, monitoring, debugging and maintaining a HDS. For an effective support of these activities, a conceptual view of a distributed system is needed. We mean by conceptual view the specification of the system as seen by the community of developers, in terms of structure and behavior. It should be noted that existing environments (or "middleware") do well in specifying structure, i.e., attributes and structural relationships among system components. However, in terms of behavior, the specification is limited to the definition of function or method signatures. The semantics of behavior are hidden or "buried" inside the application code (implementation). A high-level specification model, such as that of UML, Zoom or HeNCE (or extensions to them) is needed to formally define, at a high-level the structural and behavioral semantics of a HDS. The advantages of a formal model are twofold: it allows the automated or semi-automated translation into an implementation, and it provides a common reference model for different activities of the development process (specification, coding, debugging, etc.) An integrated development environment (IDE) should include tools for the efficient specification (design and implementation), debugging, monitoring and maintenance of HDS applications. In other words, the IDE provides a framework that supports all the activities in the development process of HDS applications.

3. RELATED WORK

Related works include HDS support environments and tools, event services and event-rule approaches.

3.1. HDS Support Environments and Tools

Many commercial HDS support environments are already available, supporting standards such as CORBA [OMG98], DCE [Open97], Java RMI [Sridhar97] and DCOM [Sessions97]. All of them have in common a “programmatic” view which lacks abstractions, tools and services for specifying, monitoring, debugging and maintaining HDS under an integrated development environment.
Bates’ work on EBBA (Event-Based Behavioral Abstraction) led to the development of a toolset for specifying and debugging distributed systems [Bates95]. An event description language (EDL) is used to define simple and composite events, and behavioral models of different scenarios of a distributed system. EDL expressiveness is limited to specifying the expected behavior of a system for validation purposes. It does not have an object-oriented model. However, the concepts of this work have inspired part of the proposed research.

Zoom and HeNCE [Wolski95] are representations to specify the structure and behavior of a HDS. Zoom has a three-level notation for specifying structure, algorithms, and data conversions. Zoom notations are translated into HeNCE, which consists of a graph-based notation of data and control flows. Neither Zoom or HeNCE are based on the notion of events, and their behavioral abstractions are procedural. The proposed research on the IDE component will study the possibility of extending these representations to include the more declarative event/rule paradigm.

Recent works on HDS support environments have focused on High Throughput Computing (HTC). Efforts in this direction are the Globus [Forster98], Condor [Litzkov88] and Legion [Grimsaw96] projects. The Globus project is closely related to ours in the development of monitoring tools; however, there is no event service or a common event-rule specification approach. In Condor and Legion, the emphasis has been in the development of algorithms and architectures for HTC by efficiently managing shared distributed resources. We foresee the proposed research as an enhancement of these projects. For example, ERF can be used for developing resource sharing support systems.

3.2. Event Services

CORBA (Common Object Request Broker Architecture) [OMG98] has incorporated events and event services in its object model [Schmidt97]. The modules CosEventComm and CosEventChannelAdmin have a set of services that support event management. These services support different types of event propagation models (canonical push, canonical pull, hybrid push/pull, hybrid pull/push). In all four models (and contrary to our approach), event handling is performed at the object level, i.e., only localized event handling is supported. Composite events and delayed event handling are not supported.

In the work by Bacon et al [Bacon95], CORBA's Interface Definition Language (IDL) is extended to incorporate events. Like in our proposed work, events are uniformly treated as objects using an event class hierarchy. Simple registration and notification services are provided, relying in client-supplied event handlers. Only a limited set of composite event types is supported by means of a language based on finite state machines.

DCE (Distributed Computing Environment) [Open97] has not yet incorporated events or events services in its object model. However, Cohen and Wilson [Cohen96] proposed an event management service based on the CORBA event specification. Only event-attributed-based event filtering is proposed. Composite events are not supported in their proposal.
Mansouri and Sloman present a configurable event service for the Darwin distributed environment [Mansouri96]. Their approach is similar to ours in the use of rules as an abstraction for the specification of event handling. A language called GEM is used to define events and rules for event handling. Many of different types of composite events are supported by a set of predefined operators. Temporal constraints are also supported. Contrary to our approach, events, rules and the event service are not uniformly treated as objects.

EVEREST [Spezialetti95] is a system tailored for the study of various approaches to event recognition and notification. It allows sophisticated users to define, by means of a script file, primitive and higher-level (i.e., composite) events, and assign event handling to predefined processes. A set of operators are defined to capture the semantics of composite events. Contrary to our approach, event handling responsibility is relied to predefined processes called monitors. The computational model of this system is not object-oriented.

ISIS [Birman94] is a CORBA-based toolkit for supporting distributed messaging in form of events. It provides a C library for support services such as channels, multicast, membership management, failure detection, fault-tolerance, and group monitoring. The main purpose of ISIS is to provide an environment for group messaging. Event processing is done at the client side (client-supplied event handlers).

3.3. Event-Rule Approaches

Most of existing event-rule approaches have been applied to active databases. Examples of such approaches are Ariel [Hanson92], Sentinel [Chak94], REACH [Buch95], OSAM*.KBMS [Su93]. They all have in common the use of a rule language for specifying database operations to perform upon the occurrence of events, which are other database operations. The proposed work is an extension of this concept to support HDS. Sentinel and REACH, although applicable to active databases, are more general extensible object-oriented systems. Although they were not designed specifically for supporting HDS, they are similar to the proposed research in the treatment of events and rules as objects in a rich object-oriented model. However, rules are compiled and tightly coupled with the database system.

Su et al. proposed the NCL language [Su96] and an Knowledge Base Management System (OSAM*.KBMS) [Su93] which use rules as abstractions for specifying interoperability in HDS. NCL is a combination of the EXPRESS and IDL languages proposed to be used in the NIIIP project, using CORBA as the underlying support environment. OSAM*.KBMS does the rule processing that is not supported in CORBA. NCL and OSAM*.KBMS do not treat events or rules as objects, and only simple events are supported.
4. DEVELOPMENT AND EXPERIMENTATION

In the following sections, we describe in detail the development and experimentation activities, together with the research issues to be addressed.

4.1. RUBIES: A Rule-based Intelligent Event Service

**Investigator:** Javier A. Arroyo-Figueroa

4.1.1. Overview

We propose the development of a Rule-Based Intelligent Event Service (RUBIES) which will be the engine with the functionalities required to support the event-rule framework. The design of RUBIES will be driven by the following rationale:

**Object-Oriented.** We will follow the object-oriented paradigm supported in environments such as CORBA, DCE, DCOM and Java RMI. Events, rules and the RUBIES itself are to be treated as objects.

**Robust event/rule model.** The event and rule models of RUBIES support simple and composite user-defined events, relative timing and relative presence/absence. Composite events are defined by expressions which can be nested at any level. Events are uniformly treated as objects and defined by an event class hierarchy, thus allowing extensibility.

**Rule-based event handling.** Instead of being buried inside application code, event handling is performed by means of rules. ECAA rules (Events-Condition-Action-AlternativeAction) are defined in terms of event filtering specification (triggering events, priority, conditions), actions and alternative actions.

**Rule scheduling.** Rules can be scheduled to be active or inactive at different points in time. RUBIES will provide the functions for allowing scheduling of rules.

**Immediate and delayed event handling.** Both immediate and delayed event handling is supported by keeping events in an event queue during a specific period of time (“time-to-live”) defined for each event. Delayed processing is carried out by inactive rules which, upon activation, are processed against events waiting in the queue.

**Distributable and replicatable architecture.** The architecture of RUBIES allows for the distribution of load among different instances of the service (for performance). Similarly, many replicated instances of RUBIES may co-exist in a HDS (for fault tolerance).
4.1.2. Distribution architecture

RUBIES follows the conventional distributed system architecture, where many different objects corresponding to distributed services communicate through a network by means of message passing. The architecture of RUBIES allows for the distribution of load among different distributed (for performance) and/or replicated (for fault tolerance) event services. This is achieved by allowing many instances of RUBIES to co-exist in a HDS. Each service that is a client of RUBIES may interact with one or more of its instances by invoking methods from its interface. The interface of RUBIES includes methods for registration, rule definition, scheduling and event notification (posting).

4.1.3. Event model

In RUBIES, events are uniformly treated as objects. An event type defines the structure and behavior that is common to a set of like events, and is represented by an event class. The class Event is the base class of all event types. It defines the structure and behavior that is common to all event types. Figure 2 presents a Java language definition of the class Event, which is a result of our preliminary studies.

```java
class Event extends Object {
    // event attributes
    private TimeStamp daytime; // event day-time (when it was produced)
    public int ttl; // time-to-live
    public Object producer; // who produced (posted) the event
    // event methods
    // constructor
    public Event(TimeStamp daytime, int ttl, Object producer);
    // return the time the event was produced
    public TimeStamp t();
    // get the name of the class of event producer
    public String getProducerClass();
    // get the producer object of the event
    public Object getProducer();
    // return true if event is dead (time to live expired)
    public boolean isDead();
}
```

Figure 2. Event class specification

Each event is uniquely identified by an object identifier, which is inherited from the class Object, which is assumed to be the base class for all objects in the system. When an event is produced (posted) the attribute “daytime” is updated with the value of the day and time when it was produced. We assume that synchronized global time is used. The value of attribute “ttl” (time-to-live) is the number of time units that the event will be kept alive. The type of time units is assumed to be a system parameter like milliseconds, microseconds, etc. A value of null in ttl indicates that
the time to live is undetermined, thus the event will be “killed” by other means such as the event service. The event “producer” is the object which produced the event. Methods of Event include its constructor; the method “getProducerClass”, which returns the name of the class of the event producer, and the method isDead() which returns true if the time to live is expired (that is, current_daytime - daytime > ttl).

In Figure 3, an example of an event type specification is presented. In this example, the event type PrecipReport is defined by means of an event class of the same name. Notice that in addition of the inherited attributed “time”, “ttl” and “producer”, this event type defines new attributes such as “p” and “Sensor”, corresponding to the amount of precipitation and the sensor which reports precipitation in a flood alert system.

An event occurrence is notified (produced) by posting an event. An event is posted by creating an instance of an event class and notifying RUBIES of the occurrence of the event by invoking the postEvent() method. When an event is posted to a RUBIES instance, the event is consumed by firing all applicable rules. The events is kept into a queue until it dies (i.e., its time-to-live has expired) or its time in the queue exceeds a predefined threshold.

```java
class PrecipReport extends Event
{
    // event attributes
    public float p; // amount of precipitation reported
    public Sensor s; // which sensor reported the precipitation
    // event methods
    // constructor
    public PrecipReport(TimeStamp daytime, int ttl, Object producer,
    }
```

Figure 3. Specification of the EMailReceived event type

4.1.4. Rule model and specification language

Rules are an abstraction that allow to declaratively specify the behavior of a HDS. Our preliminary studies show that a rule model should incorporate the following:

**Trigger events.** The trigger event specification is a set of events that will trigger the execution of the rule. A simple event triggers the rule upon its occurrence. Composite events trigger the rule depending on the relationships specified among events (e.g., one event along with others, one event but not others).

**Usage.** Sometimes rules need to make use of existing services of an HDS (e.g., functions from a math library). The usage component of a rule specifies which services are to be used in evaluating a condition or performing an action.
**Priority.** The execution paradigm of rules needs to include rule priority, such that semantics are captured correctly by executing rules in the appropriate order.

**Condition.** A condition may be specified for a rule, which must be satisfied ("true") to execute the rule upon the occurrence of events. Rule conditions may include time relationships among events (e.g., one event before another, one event 5 minutes after the other).

**Action.** The action specification of a rule states a list of operations to be perform if the trigger events occur and the condition is satisfied. This is the event handling part of the rule.

**Alternative action.** A list of operations can be specified to be performed when the rule is triggered but the rule condition is not satisfied.

The proposed rule specification language will have a syntax like the following:

```
rule <rule_id>
[ priority <priority_no> ]
on <trigger_events>
[ use <usage_specification> ]
[ if <condition> then ]
do <actions>
[ otherwise <alt_actions> ]
```

Examples of rules are given in the Validation and Demonstration section of this proposal.

**4.1.5. Interface and behavior of RUBIES**

The interface of RUBIES will have the following functionalities:

**Registration:** The interface will provide means of registering four types of clients: (i) *managers*, which are processes authorized to use configuration and management methods, (ii) *event producers*, which are services which notify the occurrence of events; (iii) *event consumers*, which are services to which the notification of event occurrence can be forwarded; and (iv) *facilitators*, which are services used by the event service to perform condition evaluation and event handling.

**Configuration.** Configuration methods will allow authorized clients (managers) to configure system parameters like default time-to-live and event queue size.

**Rule management.** Methods will allow authorized clients to manage rules. New rules can be added, existing ones can be modified, and a rule evaluation schedule can be defined.
dynamically, and such changes should take effect immediately. For each rule object, a set of attribute values are set, such as: trigger events, priority, conditions, actions and alternative actions.

**Notification.** Notification methods will allow event producers to notify the occurrence of events. Upon notification, applicable rules that are in the schedule are evaluated and executed.

**Event forwarding.** Events can be forwarded to registered event consumers. In this case, it is assumed that the consumer does the event handling, and the event service acts as an event filter.

### 4.1.6. Research issues

The following issues will be addressed in our research:

**Design of event/rule model and specification languages.** A formal specification and computational model for events and rules is needed in order to do an effective translation into a design and implementation of a support system. Similarly, a specification language for events and rules are needed to provide high-level abstractions more understandable by humans. The design of such models and languages is driven by the modeling requirements of diverse HDS application domains. This research aims to identify the common requirements for modeling and specifying events and rules in an event-rule framework for supporting HDS, and design the model and the specification languages based on such requirements. The research is going to be influenced by the requirements of the Validation and Demonstration component of this proposal.

**Design and implementation of RUBIES.** A Rule-Based Intelligent Event Service will be the heart of our Event-Rule Framework. Based on an event/rule model and the functional requirements, RUBIES will be designed, implemented and tested in the ERF environment. Design issues include the need to support a variety of existing environments such as CORBA, DCOM and DCE; and the design of a virtual machine for rule processing. Implementation issues include efficient rule firing, event pattern matching and mapping to language- and environment-specific implementations.

**Optimization of rule processing in RUBIES.** We visualize the use of RUBIES for supporting HDS that require real-time performance. Rule processing poses an overhead caused by event pattern matching and condition evaluation. On the other hand, the declarative nature of rules gives an opportunity for optimization. This research aims to discover optimization algorithms for rule processing.

**Replication of RUBIES for fault tolerance.** Many mission-critical applications cannot afford shutdowns. Sometimes, such shutdowns are due to the failure of a single system component. Many replicated and coordinated instances of RUBIES will reduce the probability of shutdowns due
to the failure of a single instance of RUBIES. This research aims to design an interface and discover efficient algorithms to support replication.

4.2. An Integrated Development Environment

Investigators: Javier A. Arroyo-Figueroa, José A. Borges and Néstor Rodríguez

4.2.1. Objectives

An integrated development environment (IDE) should include tools for the efficient specification, coding, debugging, monitoring and maintenance of HDS applications. In other words, the IDE provides a framework that supports all the activities in the development process of HDS applications. Our goal in this research is to develop an IDE that will support the requirements specification, design, coding, monitoring and debugging processes. The proposed product has the following design rationale:

**Easy to learn and use.** Human computer interaction and usability engineering issues and methods will be given high priority to make sure that this goal is accomplished. A unified model to define rules and events will be developed such that a consistent representation could used on most aspects of the HDS life cycle.

**Language independence.** User should not be required to know a specific programming language (e.g., C++, Java) in order to specify the structure and behavior of a HDS at a high level.

**Independence of the underlying HDS environment.** The development of an HDS using this environment should be independent of the underlying environments (CORBA, DCE, DCOM, Java RMI, etc.). In fact, users should be able to specify or select any underlying environment.

4.2.2. Proposed tools

An HDS framework as defined in this proposal requires tools to support the following processes:

**Specification.** The framework will also include tools to specify application, event and rule objects. This will be useful for supporting the requirement specification, design and coding processes. The application objects in an HDS will be the services taking an active part in the distributed application. In order to communicate with the services from the distributed systems as objects it is necessary to define them with wrapper classes. Wrappers enclose the attributes and interfaces of the services. The framework should provide a tool to select a wrapper category, associate it with a specific service, and specify its attributes and methods. This process will generate code according to the a particular HDS environment (e.g., CORBA). Similar to application objects, the event objects can be defined based on a standard protocol. An event category is
selected from the Event hierarchy, or a new category is defined, then their attributes and methods are specified. The definition of composite events is based on the rule model. Rule objects are used to specify system behavior based on the ECAA (Event-Condition-Action-AlternativeAction) paradigm. The tools should include facilities for specifying rules.

**Monitoring and debugging.** These two processes are highly coupled. Monitoring and debugging tools are needed to keep track of event occurrences, event flows, message flows, rule triggers, event forwarding and changes in the state of the system. These tools are needed for both simulation runs and during real-time execution of applications.

Figure 4 presents the run-time and build-time scenarios of usage of the IDE. It is convenient to present both scenarios together, since the IDE will allow to define events and rules dynamically. Build-time flows are represented by dashed arrows; run-time flows are represented by solid arrows. During build time, developers may define wrappers, events and rules, which go into the IDE repository. Notice that the IDE repository might be a distributed system component accessed by the HDS communication infrastructure (e.g., CORBA’s ORB). The middleware compiler converts the language-independent wrapper specifications into language-specific wrappers, which are used to wrap distributed system components during build time. During run time, the specification tool can
be used to specify new or modify existing events and rules. RUBIES instances will access the IDE repository to obtain event and rule specifications. System components generate data and events. Events are sent to RUBIES instances, which process the applicable rules, which in turn may generate other events or perform method invocations. The monitoring and debugging tool is used by developers to generate events and data which may be sent to other system components or instances of RUBIES. Monitoring data generated by RUBIES is used by this tool to display system status to the user.

4.2.3. Research issues

In addition to the design, implementation and human-computer interaction issues implied in the design of tools, our research will address the following issues:

**High-level representation.** A graphical high-level model, such as that of UML [Booch98], Zoom [Anglano95] or HeNCE [Wolski95] (or extensions to them) needs to be developed in order to represent the structural and behavioral semantics of a HDS based on abstractions of the objects developed or defined using the above tools. This model is critical because it will be the basis of other tools for the specification, design, implementation, debugging, monitoring, and maintenance of the system. Also, it the basis for translation into many different HDS support environments.

**Visual languages.** One of the major challenges of this research will be determining how to define, represent, debug, and update rules in a consistent, easy to learn, and easy to use fashion as stated in the above objectives. A possible solution to be researched is to define a visual programming language for the rule model. A visual language will not only provide a consistency in the model which could be used for all those aspects mentioned, but could also give us the following potential benefits: (i) language independence, (ii) higher information transfer rate, (iii) an intuitive and clear model, (iv) a representation that could be set in motion to show dynamics directly, (v) direct access to objects by direct manipulation, (vi) a significant reduction in the number of syntax details, (vii) effective feedback at all stages of development, (viii) access to different representations or abstraction levels of an object, (ix) improved user productivity, and (x) reduced learning curve. There are many limitations and problems related to visual languages as well. The major concerns regarding our research are:

- Visual languages have difficulty with large programs since their representation are usually larger than the text they replace.
- Traditionally, visual languages have been more effective in a limited domain, with a limited set of data types and built-in operators
- There is no guarantee that people can interpret the meaning of a picture as intended by the designer.
- A visual programming language is more dependent on the user interface than the traditional text based languages. The visual language will be only as good as the interface on which it is presented.
These issues will determine if a visual programming language is appropriate for the rule model. Since the extent of programming for this application are the specification of individual rules and since rules are based on an ECAA protocol, the main concern will be the scope of the relations between events, the scope of the operators needed to specify the triggering conditions, and the levels of abstraction that can be reasonably represented without degrading the programming or specification process performance. An alternative compromise would be a hybrid language (visual and textual) or languages based on forms or templates. Using some of these variations we could still achieve some of the benefits mentioned above.

4.3. Validation and Demonstration: Real-time Flood Alert System (RTFAS)

Investigators: José A. Borges and Néstor J. Rodríguez

The development of a Real-Time Flood Alert System (RFTAS) has been chosen to validate the ERF because it has ideal characteristics for an ERF implementation. This system: (i) comprises many applications which reside in dissimilar platforms, (ii) requires the processing of many simultaneous events, (iii) requires real-time processing of events, (iv) requires the processing of different sets of rules which may involve multiple distributed event services, (v) needs continuous monitoring for debugging and validations purposes.

A description of the existing alert system is presented in the following section. The proposed RTFAS system is described in Section 4.3.2. Some examples of the applicability of ERF to the implementation of RTFAS are presented in Section 4.3.3. In Section 4.3.4, the proposed development methodology for the user interface is described. The procedure for validating the system is discussed in Section 4.3.5.

4.3.1. Current system

Flood detection and alert in Puerto Rico relies on a network of gaging stations and a Doppler radar (see Figure 5). The US Geological Service (USGS) and the Civil Defense of Puerto Rico maintain the gaging stations. The USGS maintains 123 near real-time gaging stations for measuring river stage (height) and discharge (volume), lake elevation, and precipitation. These stations transmit data via satellite to a local USGS office. During normal operation the stations transmit every 4 hours. Under storm conditions when a sensor detects a rapid rise in precipitation or in the stage of a river, the station goes into an alert mode and transmits every 5 minutes. The Civil Defense maintains 40 radio transmission stations for monitoring precipitation. The National Weather Service maintains the Doppler radar. This radar generates color images of the Puerto Rico vicinity that provide information on the areas where there is precipitation. The intensity of the precipitation is indicated by means of a color scale. This information is used by the National Weather service to calculate the instantaneous and accumulative amount of rainfall. The information generated by the staging stations and the Doppler radar is made available to the Aqueduct and Sewer Authority (ASA), the Puerto Rico Electric Power Authority (PREPA), the Civil Defense (CD), and the National Weather Service (NWS).
The information received by the ASA is used for the early detection of high influx into dams. In anticipation of a high influx the level of the lake is lowered to prevent a dam overflow. Changes in the measurements of the different stations constitute events that can trigger four alert stages, namely risk watch, moderate risk, high risk and event-in-progress. The information provided to the ASA is mostly raw. Thus, most of the events are detected by calculations performed by operators of the dams.

The NWS uses the data provided by the precipitation monitors and the Doppler radar to issue flood alerts. The following three alerts can be issued: (i) flash flood watch, issued when current and developing hydro meteorological conditions are such that the area designated in the watch message is subject to possible flooding, but the occurrence is neither certain or imminent; (ii) flash flood warning, issued for specific communities, streams, or areas where flash flooding is imminent or in progress; (iii) flood warning, issued when inundation of a normally dry area near a major water course or ponding of water is expected. The type of flood alert and the time it is issued is at the discretion of a weather forecaster.

The CD uses the information provided by the precipitation monitors and the Doppler radar to generate flood alerts for the municipalities. This agency does not have predefined procedures to generate alerts. The Electric Power Authority uses the information of the lake elevation measurements to control the level of the dams administered by them.
In general, the existing alert systems of the different agencies lack automation and integration. Many applications work independently in dissimilar computing platforms (PC/Windows, PC/QNX, Sun/Solaris). The data is provided to the agencies without much processing and is usually displayed in the form of tables. Most of the events are detected and the alerts generated as a result of human calculations and judgment. The user interfaces are mostly character-based and lack usability. The format in which the information is presented does not necessarily correspond to the user's need. When several, simultaneous events take place it is very difficult for the users to get a precise interpretation of the status of the rivers and lakes.

From an analysis of the current system we are convinced that a more effective and useful flash flood alert system can be developed following an event-driven implementation approach and by applying usability engineering principles to the user interface design. The proposed system is described in the following section.

4.3.2. An event-driven alert system

The proposed Real Time Flood Alert System (RTFAS) will receive the data provided by the gaging stations of the U.S. Geological Survey and the Civil Defense, and the images of the Doppler Radar of the National Weather Service. This data will be processed to detect events and report hydrological data and alerts to the Aqueduct and Sewer Authority, the Puerto Rico Electric Power Authority, the Civil Defense and the National Weather Service (see Figure 6). This information will be used by the different agencies for forecasting flash floods, monitoring rainfall events, and reservoir management control.

RTFAS will be developed as an event-driven system. The most basic events will be the reporting of the measurements of a gaging station and the arrival of a Doppler radar image. A combination of events can determine other events. For example, a difference of more than .25 inches between the last two measurements of a precipitation monitor could constitute an event. A single event or a combination of events could trigger an alert. The kind of alerts and the events that trigger them will be determined by each of the agencies that are served by the system. The alerts will be transmitted to the interested agencies.

This development will be carried out in three major cycles. The first cycle will involve (i) the specification of requirements from both the application perspective (the product) and the development process perspective (the process); and (ii) the design of the user interface. In terms of the product, functional and data requirements will be analyzed and specified. Many functional requirements will be specified in terms of events and rules. This will serve to identify modeling requirements (the process), which will influence the design of the event and rule models and specification languages. In the second cycle, a prototype of RTFAS will be developed with the available prototypes of RUBIES and tools. This cycle will be useful for providing feedback to the RUBIES and IDE development groups. In the third cycle, it is assumed that a prototype of ERF is available, and the ERF IDE itself will be used in the development of the system, including specification, design, monitoring and debugging. This will validate and demonstrate the use of ERF in the development of HDS.
4.3.3. Event and rule examples

The following are examples of events and rules that show how promising will be ERF for the development of RTFAS.

Risk Watch: The Aqueduct and Sewer Authority is interested in this type of event. Such event happens when a weather system capable of generating heavy rainfall that could result in high influx into a lake is approaching a specific zone. A rules is needed to generate this type of event. The rule will be triggered when two events of type WeatherSystemMoved have occurred:

```java
// Event generated by Weather Image Analysis System (WAIS)
class WeatherSystemMoved extends Event {
    int wsid;        // weather system identifier
    float lat;      // latitude
    float lon;      // longitude
    float level;    // percentage of rain concentration (doppler)
}
```
class RiskEvent extends Event {
    Area a; // the area for which the risk is involved
    int level; // level of risk (watch, moderate, high)
}

// Generate risk watch event if weather system is near and moving towards area
rule riskWatch1
on WeatherSystemMoved wsm1 && WeatherSystemMoved wsm2
use Area a, CartesianSvc cs
if wms2.t() > wms1.t() && wms1.wsid == wms2.wsid && wsm2.level > 0.6
    && cs.distance(a.lat,a.lon,wms2.lat,wms2.lon,MILES) <= 10
    && cs.crossesCircle(wms1.lat,wms1.lon,wms2.lat,a.lat,a.lon,a.radius)
then do
    post RiskEvent(a,WATCH)
end

The event WeatherSystemMoved is assumed to be generated by the WAIS (Weather Image Analysis System) service object. This service is assumed to be a sophisticated image analyzer capable of detecting the storms. This rule is triggered when two events (wms1, wms2) of this type occur. The rule makes use of two distributed objects: an Area object, corresponding to an area registered to be of interest for the event, and a distributed service called CartesianSvc, which includes methods related to operations on a Cartesian plane. In this case, the methods “crossesCircle” and “distance” of CartesianSvc (cs) are used in the “if” condition of the rule. Notice that the rule’s condition compares the time of occurrence of the two events by invoking the t() (time) method. The condition also checks that the two events correspond to the same weather system, by comparing their wsids. Also, the rule checks that the level reported in the second event exceeds a threshold (0.6). In addition, the “distance” method of “cs” is used to get the distance between the two points (the area and the center of the weather system), to check if it is near (<=10 miles). Finally, it checks that the projection according to CartesianSvc says that the imaginary line will cross the circle defined by the area’s coordinates and radius. If all conditions are true, then a new event (RiskEvent) is posted. Notice that the parameter to the event is the area “a”.

The previous example demonstrated that events can generate other events. A derived event can be used in other rules. The ASA is interested in generating an alert event if a risk event happens in the Carraizo area. The following rule will be triggered when a RiskEvent occurs, and uses the CarraizoAlertSystem to report an alarm only if the area is Carraizo:

rule carraizoAlert1
priority 10
on RiskEvent re
use CarraizoAlertSystem cas
if re.a.name="Carraizo" then do
cas.alarm(re.a.level)
end

High Risk: A high-risk event occurs when an isolated precipitation monitor reports more than .5 inches in less than 15 minutes. This can be captured with the following rule, which is triggered by when two events of type PrecipReport (see Figure 3) occur:

rule highRisk1
on PrecipReport pr1 && PrecipReport pr2
if abs(pr1.t() - pr2.t()) < 15 && pr1.s.id = pr2.s.id && abs(pr1.p-pr2.p) > 0.5
then do
    post RiskEvent(pr1.s.area,HIGH)
end

This rule checks if the time difference between the two events (pr1 and pr2) is less than 15 minutes, the two sensor id (pr1.s.id, pr2.s.id) of the two events is the same, and the difference between the two precipitation reports exceeds 0.5, a new RiskEvent will be posted to the system, for the area associated with the sensor (pr1.s.area), with a HIGH risk level.

More sophisticated rule constructs will be needed to support all the system modeling requirements. Such issues will be addressed in this research.

4.3.4. The user interface

The user interface is a critical aspect of the proposed RTFAS system. A user interface that is not easy to use and that does not provide useful information is not going to be too useful and will hide the advantages of an event-driven system. The objective of the user interface is to provide the information the user needs at exactly the time he/she needs. The information should be precise and easy to understand. To accomplish this objective we must conduct an in-depth task analysis with the personnel that deal with the alert system on the different agencies that will be served by the system. This task analysis will help us identify the kind of information that is need in each of the agencies. Then, we must apply usability engineering principles [Nielsen93] in the design of the user interface. Various usability engineering techniques will be employed through the design process. During the early stages the user interface prototype will be subjected to usability heuristic evaluations [Nielsen94]. At later stages the think aloud technique will be employed. At the final stages a formal test with real users will be conducted.

4.3.5. System validation

System validation will be accomplished by conducting a formal test with the personnel that should deal with the alert system. For this test we will collect data from the gaging stations and the Doppler radar during the passage of various weather systems that generate floods. Each data element will be time-stamped. We will also collect information on the alerts issued by the
operators of the different agencies during each weather system. The kind of alert issued and the time it was issued will be recorded.

The data collected from the gaging stations and the Doppler radar will be fed into the system. In turn, the RTFAS system will generate the appropriate alerts that will be interpreted by operators of the different agencies. The monitoring tool will be used to track the generation of events and the triggering of rules. The time when the operator recognize the alert will be recorded and eventually compared with the times originally recorded during the actual passage of the weather system. The time differences will serve as a measurement for comparing the effectiveness of the proposed system and the existing system.

To validate the user interface, at the end of the test the operators will be interviewed, to get their impressions (user satisfaction) of the new system in comparison to the existing one. In particular we would like to find out whether they felt that the system: was easy to use, provided precise information, did not lack information, responded quick enough.

5. RESEARCH INFRASTRUCTURE

To ensure the success of this project, and with the intention to enhance the research infrastructure of the Electrical and Computer Engineering Department at UPRM, we request funding for the creation of a Heterogeneous Distributed Systems Laboratory (HDSL). HDSL’s primary goal is to provide facilities for research and development of tools, algorithms and environments for supporting HDS. The laboratory will bring a working environment which will promote the collaboration among researcher and give access to the necessary hardware, software and environment for successfully carrying out research on HDS. Since it would not be practical to have all existing platforms in a single HDSL, we propose to have a subset that is representative of state-of-the-art technologies. The laboratory will have the following elements:

Servers. The laboratory will have three servers with the following platforms: Sun/Solaris, Pentium/Solaris and Pentium/Windows-NT. These servers are needed to have a multi-platform environment representative of existing technologies.

Workstations. Six Pentium/Windows-NT workstations will be used by researchers for the research and development process.

Network. Networking is a fundamental component in HDS. All computing resources of the laboratory will be connected in a 100-Base-T network. The network will be connected to ECENet (ECE department network) and the internet.

Development tools. The laboratory will include software development tools supporting multiple languages such as Java, C++ and C. Rational Rose, Inprise JBuilder, Microsoft Visual Studio will be included.
**HDS middleware.** To do experimentation with multiple standards, it is important to have installed in every platform multiple HDS middleware. Each machine will have installed environments supporting standards such as CORBA, DCE, DCOM and Java RMI. We plan to purchase BEA ObjectBroker (which supports CORBA and DCOM), Digital DCE (for Windows NT) and BEA Tuxedo (DCE for Solaris). Java RMI is included in the Java run time libraries.

**Database management system.** Database systems are an important component of any support environment. An Oracle database system will be available in the Sun/Solaris server.

System administration of the laboratory resources will be provided by the ECE department’s system administrators.

**6. INTEGRATION WITH EDUCATION**

We believe that an educational plan integrated with the proposed research is critical for the complete success of this project. We propose a set of activities that will enhance education through (i) development of undergraduate research projects, (ii) enhancement of graduate curricula, (iii) supervision of graduate and undergraduate students, and (iv) dissemination of research.

**Development of undergraduate research projects.** The undergraduate programs at our institution will be enhanced by giving undergraduate students the opportunity to perform research. Although some implementation issues of the proposed research might be trivial from the research perspective, they offer a valuable opportunity for senior undergraduate students to (i) apply the knowledge they have acquired in the development of a product, (ii) enhance their knowledge and capture their interest in graduate studies by working in a project related to a major research problem. We expect to hire three undergraduate research assistants per year.

**Enhancement of graduate curricula.** In order to capture the students’ interest in the proposed research, it is necessary to expose them to the fundamentals of the field. We plan to accomplish this through two graduate courses: (i) Human-Computer Interaction (HCI) and (iii) Distributed Operating Systems. The course in HCI (prepared by Drs. Borges and Rodríguez) will expose the students to the fundamentals of design and implementation of user interfaces. We plan to use the proposed project as a case study of HCI. The course on Distributed Operating Systems (prepared by Dr. Arroyo) will enhance the students’ knowledge of distributed systems, environments and algorithms. One important subject in both courses is an overview of current research topics, which will include the proposed research.

**Supervision of graduate and undergraduate students.** Although this is an obvious activity of any educator, we stress its importance here because we believe that it should work together with an educational philosophy. Such philosophy includes: (i) the proper training of students in the understanding of the research problem and the use of the research infrastructure, (ii) periodic information and “brainstorming” meetings to improve understanding and the sharing of ideas, (iii) the assignment of reading of related publications and establishment of relationship with
current work, (iv) the request of periodic writing of reports to assess their understanding, and (v) the
evaluation and feedback to the students to improve their writing skills.

**Dissemination of research.** Dissemination activities will enhance education by giving
other students the opportunity to know the purpose, status and products of our research. Such
activities include: (i) publication and maintenance of a project web page, (ii) writing of conference
and journal papers, (iii) presentation of our work in local, national and international conferences
and workshops, (iv) stimulate students to participate in such conferences and workshops, and (v)
seminars to present periodically our work at local and national universities.

7. RESEARCH AND EDUCATION PLAN

7.1. Project schedule and milestones

The project is expected to last three years. It will start on Summer 1999 and end on Spring
2002. The following table presents the milestones of the project.

<table>
<thead>
<tr>
<th>Semester</th>
<th>Deliverables</th>
</tr>
</thead>
</table>
| Fall 1999      | HDSL installed  
                 Research assistants hired and trained  
                 Graduate courses Spring’2000 preparation  
                 RTFAS requirements specification  
                 Event-Rule model |
| Spring 2000    | RUBIES design  
                 High-level spec. model  
                 RTFAS design |
| Summer 2000    | Specification tool design  
                 Monitoring and debugging tool design |
| Fall 2000      | Graduate courses Spring’2001 preparation  
                 RUBIES implementation - version 1 |
| Spring 2001    | Specification tool implemented - version 1  
                 RTFAS prototype  
                 Monitoring/debugging tool implemented - version 1 |
| Fall 2001      | RUBIES implementation - version 2  
                 Specification tool implemented - version 2  
                 Graduate courses Spring’2002 preparation |
| Spring 2002    | RTFAS implemented  
                 RUBIES implementation - version 3  
                 Monitoring/debugging tool implemented - version 2 |
Table 2 presents a schedule of the following proposed activities. If an activity spans during a given semester (Summer, Fall or Spring), the corresponding box is marked with an “X”. When a deliverable is due, the corresponding box is marked with a bold X. The following are the research and education activities:

**Development of HDSL.** This activity includes the purchasing of equipment and furniture, and the construction activities for the creation of the Heterogeneous Distributed Systems Laboratory (HDSL). We expect HDSL to be ready by Fall 1999.

**Graduate course development and enhancement.** As proposed in the plan for integration with education, two graduate courses will be developed, namely Human-Computer Interaction and Distributed Operating Systems. This activity will be repeated each Fall semester to improve the two courses by integrating material derived from the proposed research.

**Hiring and training of students.** Potential graduate and undergraduate research assistants will be interviewed. The best candidates will be hired and trained on the research problem. Periodic meetings will be held to keep track of their progress during the training period. The investigators already have one graduate student each (doing independent study in Spring 1999), for starting training on Summer 1999. This activity will be repeated on each Fall semester.

**Undergraduate research projects.** We expect to support three undergraduate research assistants. Their projects will consist of the implementation of minor parts of the system, e.g., parts of the user interface, specific methods of RUBIES, performance monitoring of a subset of the system or simulations. This activity will be repeated every regular semester.

**Development and maintenance of project web page.** As part of our dissemination plan, we perform this activity which will span many semesters. An undergraduate student will be hired part-time to perform this task.

**Paper writing.** Also part of the dissemination plan, we will be writing papers to be presented in conferences, workshops and journals. Candidate conferences are OOPSLA, CIKM, ICDCS, IPPS and CHI. Candidate journals are JIIS, ACM TOCHI, IEEE TKDI and IEEE Software.

**Design of event and rule model.** This is an iterative activity which will be influenced by multiple-domain analysis performed by investigators, as well as by the modeling needs of the RTFAS system. Since it is iterative, and thus expected to improve the model continuously, this activity spans many semesters.

**Design of high-level specification model.** An iterative activity influenced by modeling needs derived from multiple-domain analysis and the domain analysis of RTFAS. Expected to be iteratively improved during the course of this project.
Design and implementation of RUBIES. This is another iterative activity influenced by the modeling needs of RTFAS. The initial prototype RUBIES will be implemented using ObjectBroker (a CORBA-compliant environment). Later versions of RUBIES will be implemented on top of DCE and Java RMI.

Optimization of rule processing. This research activity will be performed in parallel at later stages of the project. It aims to discover optimization algorithms for rule processing, to reduce the overhead caused by event pattern matching and condition evaluation.

Replication of RUBIES for fault tolerance. This research activity will be performed in parallel at later stages of the project. It will consist of a study of architectures and algorithms that will allow replicated RUBIES instances to interact and thus avoid system shutdowns due to the failure of a single RUBIES instance.

Design and implementation of tools. The research activities related to the design and implementation of tools will be influenced by the requirements of RTFAS and the design of RUBIES. These activities include the design and implementation of specification, monitoring and debugging tools. Such activities can be performed in parallel with a strong collaboration among investigators.

RTFAS development activities. This activity consists of the requirements specification, design, prototyping and implementation of RTFAS. While the development of RUBIES and the IDE are in progress, requirements for RTFAS will be collected and specified, and a system design will be done based on the developed high-level model. Prototyping of RTFAS will be done with the partial availability of RUBIES and tools. Finally, a complete implementation of RTFAS is expected at the end of the project.
### Table 2. Project schedule

<table>
<thead>
<tr>
<th>Activity</th>
<th>Su 99</th>
<th>Fa 99</th>
<th>Sp 00</th>
<th>Su 00</th>
<th>Fa 00</th>
<th>Sp 01</th>
<th>Su 01</th>
<th>Fa 01</th>
<th>Sp 02</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development of HDSL</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graduate course development and enhancement</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hiring and training of students</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Undergraduate research projects</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development/maintenance of project web page</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paper writing</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Design of event and rule model</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Design and implementation of RUBIES</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimization of rule processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replication of RUBIES for fault tolerance</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design of high-level specification model</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design and implementation of specification tool</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design and implementation of debugging tool</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTFAS - requirements specification and design</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTFAS - prototype</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTFAS - final implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**Note:** Some activities have multiple deliverable versions. Deliverables are marked with a bold X.
7.2. Management plan

The development and experimentation activities will be carried out by three research groups, namely RUBIES, IDE and RTFAS. All groups will share the resources of the HDSL laboratory. The project manager will be Javier A. Arroyo-Figueroa, Principal Investigator.

**RUBIES group.** (Leader: J.A. Arroyo-Figueroa) This group will be responsible for the research activities and development of RUBIES. Two graduate and one undergraduate student will be working on this group. During the first half of this project, one student will work on the design of the event and rule models, and the rule language, while another will work on the design and implementation of RUBIES. In the second half, one student will work on optimization while the other will work on replication. The undergraduate student will be assigned a specific implementation task (e.g., develop the parser for the rule language).

**IDE group.** (Leader: J.A. Borges) Two graduate and one undergraduate research assistant will work in this group. They will do research and development of the specification and monitoring tools of the IDE. One student will work on the development of the specification tool, while the other will work, in parallel, on the monitoring and debugging tool. The undergraduate student will be assigned a specific implementation task (e.g., develop the IDE repository file manager). J. A. Arroyo-Figueroa will collaborate with this group.

**RTFAS group.** (Leader: N.J. Rodríguez) This group will be responsible for the development of RTFAS. One graduate and one undergraduate student will team up in this group. Both students will work closely in the implementation of the system. The graduate student, in addition to the implementation issues, will work on usability and other design issues. J.A. Arroyo-Figueroa and J.A. Borges will collaborate with this group.

The following table presents the human resource allocation for the set of major tasks of this project:

<table>
<thead>
<tr>
<th>Tasks</th>
<th>JAAF</th>
<th>NJR</th>
<th>JAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project management</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDSL management</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supervise dissemination plan</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graduate course preparation</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Supervise RUBIES group</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supervise RTFAS group</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Supervise IDE group</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**Legend:** JAAF=Javier A. Arroyo-Figueroa, NJR=Néstor J. Rodríguez, JAB=José A. Borges
8. COST SHARING BY UPRM

UPRM is committed to share costs of this project as presented in the following table:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cash</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Undergraduate assistantships</td>
<td>$9,000</td>
<td>$9,000</td>
<td>$9,000</td>
</tr>
<tr>
<td>CECORD administrative personnel (1)</td>
<td>$11,000</td>
<td>$11,000</td>
<td>$11,000</td>
</tr>
<tr>
<td>HDSL Laboratory furniture</td>
<td>$7,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDSL Laboratory remodeling</td>
<td>$3,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Networking materials and accesories</td>
<td>$3,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equipment maintenance</td>
<td></td>
<td>$7,000</td>
<td>$7,000</td>
</tr>
<tr>
<td><strong>Total cash</strong></td>
<td>$33,000</td>
<td>$27,000</td>
<td>$27,000</td>
</tr>
<tr>
<td><strong>In-kind</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Academic discharge for research (3 credits x 3 investigators)</td>
<td>$38,000</td>
<td>$38,000</td>
<td>$38,000</td>
</tr>
<tr>
<td>Laboratory space (600 sq. ft.) x $30/sq.ft.</td>
<td>$18,000</td>
<td>$18,000</td>
<td>$18,000</td>
</tr>
<tr>
<td><strong>Total “in-kind”</strong></td>
<td>$56,000</td>
<td>$56,000</td>
<td>$56,000</td>
</tr>
</tbody>
</table>

(1) CECORD is the CEnter for COmputing Research and Development of the ECE Department of UPRM. Its budget is kept separate from UPR funds.
9. EXPECTED RESULTS AND IMPACT OF PROPOSED RESEARCH

We expect that the proposed research will bring the following results:

- A framework for the development of HDS using tools based a high-level model independent of HDS middleware
- An environment and research infrastructure for continued research and education on HDS at UPRM
- A novel programming model for the development of heterogeneous distributed systems
- Extensions to existing HDS standards like CORBA, DCE and DCOM
- Extensions to current efforts on distributed systems for high-throughput computing
- Publications in journals and conferences

We expect that the proposed research will have the following impact on research and education:

**Impact on CISE research.** We expect to make a contribution to the field of HDS in terms of programming models, graphical tools, development methodologies, distributed algorithms and architectures.

**Impact on existing research efforts.** As we mentioned before, the proposed research is general enough to be applicable to existing research efforts in High Throughout Computing (HTC) like Globus, Condor and Legion.

**Impact on standards.** Our research is expected to influence existing standards like CORBA, DCE and DCOM. We believe that once the technology is proven successful, the event service can be proposed as an extension to such standards.

**Impact on UPRM graduate programs.** We are convinced that the proposed research will enhance the opportunities of graduate students to do research, by giving them the option of specializing in the areas of distributed systems and human-computer interaction. Our long-term goal is to have those as a permanent areas in our programs. The existing MSCpE program will be enhanced, as well as the NSF-CISE-sponsored PhD program which is on its way to be approved by UPRM. It is expected that eight (8) students will receive a Masters of Science degree in Computer Engineering (MSCpE) upon the successful completion of their research work on this project.

**Impact on minorities.** University of Puerto Rico at Mayaguez (UPRM) is catalogued as a minority institution. Over 90% of students are hispanic. It is our belief that the proposed research will enhance the opportunity of our institution to become involved in a research area in which minority institutions have not achieved an equitable participation.
10. RESULTS FROM PRIOR NSF FUNDING

10.1. Javier A. Arroyo-Figueroa

No prior NSF funding.

10.2. Néstor J. Rodríguez and José A. Borges

Two research projects in the area of human-computer interaction are being developed. These two research projects are being partially supported with the NSF II-MI grant entitled: "Development of a Computer Engineering Research Environment at the University of Puerto Rico at Mayagüez" (Grant CDA-9417659). The objective of the first project is development of graphical user interfaces and information systems to support the clinical activities of physicians and nurses in hospital settings. This project seeks the development of two systems: an easy to use computer-based patient record system, and a system that will keep track of the status of patients and ancillary services in an emergency room. The objective of the second project is to study how people interact with the World Wide Web in order to develop guidelines for the design of usable World Wide Web pages.

A Computer-Based Patient Record for Improving Nursing Care

Most of the hospitals in the USA and Puerto Rico keep their patient medical records in paper form. Most of the information that constitutes a patient record is entered by nurses by means of documentation forms. This documentation process takes a significant amount of the nurses time, thus, reducing the time devoted to direct patient care. In [Rodz97a] we described the development of a prototype of a computer-based patient record (CPR) system that can improve the quality of nursing care. The improvement in nursing care will be accomplished by providing mechanisms for documentation which reduce human error and the time required to accomplish it, thus, providing more time for direct patient care. The design of the system is based on the principles of human-computer interaction and usability engineering.

Our CPR system has been conceived as a series of integrated modules that will enhance functions such as processing of physicians orders, nursing documentation, laboratory reporting, display of patients’ vital sign data, assisting in the training of personnel, generation of data for outcomes research and management studies, and networking among hospitals. Our current work is focused on the modules for entering the physicians orders and nursing documentation.

On our prototype the medical orders are entered directly into a computer by the physicians through the physicians interface. These orders will be sent automatically to the corresponding departments. This releases a great deal of time from the nursing staff because the nurses no longer have to transcribe physician orders or deal with the delivery of the different forms to the corresponding departments. Thus, the nurses intervention with the patient record is reduced to the usual nursing documentation that they are required to perform. This documentation is accomplished through a nurses interface.
SAAS: Automatic System for Auto-Supervision in an Emergency Room

The development of powerful and affordable information technologies has increased the interest for clinical computing systems in many hospitals. The latest trend in clinical computing are the Computer-Based Patient Record (CPR) systems. These systems help to improve the quality of health care and at the same time reduce administrative costs. We believe that there are many possibilities to improve these systems. In [Borges97a] we presented ongoing work on the development of an automatic system for auto-supervision known as SAAS (Sistema Automático de Auto Supervisión), that introduces improvements to current CPR systems. The approach used in SAAS is based on patient flow and load behavior, and on information about resources capacity and availability. It strives to improve the flow of patients moving through the system and the assignment of resources to serve them. SAAS will be able to provide status on each and all patients as well as the load condition of each resource used in the system. As the SAAS system gains experience it will be able to automatically schedule patients and resources and send warning messages to specific administrative resources. SAAS was originally conceived and is being developed for the Emergency Room (ER) of local hospitals.

Page Design Guidelines Developed through Usability Testing

The topic of Web design has become very popular in recent years, manifested by the increasing number of books, courses, tutorials, and articles on the subject. Many guidelines for designing Web pages can be found on the WWW. Most of these guidelines are based on personal experiences and observations and are not supported by formal experimentation.

A more practical alternative for most page designers is a set of simple guidelines for designing Web pages, guidelines that avoid the common mistakes found in many pages today and that promote usability. These guidelines should be based on fundamental principles of UI design, but should be simple enough for nonexperts to understand and apply. The goal of our work has been to develop a set of guidelines that meet the following criteria: The guidelines should be short, simple, and practical; in other words, they should be easy to read and understand. Users should not need to know UI design and usability engineering to apply the guidelines, and they should be supported by usability evaluation and testing. In [Borges96, Borges97b and Rodz97b] we describe the development process and usability evaluations performed to generate a set of guidelines for designing usable Web pages.
January 8, 1999

Program Director
NSF CISE Next Generation Software Program
NSF, Room 1105
4201 Wilson Blvd
Arlington, VA 22230

Dear Gentle People:

It is with great pleasure that we commit our support for the proposal entitled "ERF: An Event-Rule Framework for Supporting Heterogeneous Distributed Systems", by Dr. Javier A. Arroyo-Figueroa (PI), Dr. Néstor J. Rodríguez and Dr. José A. Borges (Co-PIs). This project is focused in the development of an Event-Rule Framework (ERF) which comprises a set of services and tools for supporting the development of heterogeneous distributed systems (HDS). We strongly believe that this project will enhance the opportunities and the infrastructure for research in our institution.

The University of Puerto Rico at Mayagüez (UPRM), through the Office of the President, is seeking cash commitments of matching funds of $89,000. The investigators, all from the Department of Electrical and Computer Engineering at UPRM, are fully committed to this project. Concurrently, UPRM, the College of Engineering and the Department of Electrical and Computer Engineering will provide the necessary release time and physical space to support this endeavor (also accounted for in the proposal).

I sincerely hope that the evaluation team will recognize this endeavor and in June 1999 the investigators may begin the tasks depicted in the work plan.

Respectfully,

[Signature]

Dr. Zulma R. Toro Ramos
Acting Dean, Engineering School

[Signature]

Dr. Manuel Hernández-Ávila
Director, R&D Center

ZRTR:RVE
To Whom it May Concern:

Please accept this recommendation letter in support of the proposal entitled "An Event-Rule Framework for Supporting Heterogeneous Distributed Systems" by Dr. Javier A. Arroyo-Figueroa, Dr. Néstor J. Rodríguez and Dr. José A. Borges. This proposal will be submitted to the NSF Next-Generation Program.

We will actively collaborate with this research group in the development of a software application for a Real-Time Flood Alert System. Our collaboration will consist on sharing information and working together in the specification of requirements, design and prototyping of the software system, and the use of the prototype for testing purposes. In the eventual success of this effort, we will consider its incorporation into our normal operations.

We strongly believe that this product will enhance the data dissemination capabilities of our agency.

Sincerely,

[Signature]

Pedro L. Diaz
Acting District Chief

PLD/dl
22 de diciembre de 1998

To Whom it May Concern:

Please accept this recommendation letter in support of the proposal entitled "An Event-Rule Framework for supporting heterogeneous Distributed System" Dr. Javier A. Arroyo-Figueroa, Dr. Nestor J. Rodriguez and Dr. Jose A. Borges. This proposal will be submitted to NSF Next-Generation Program.

We will actively collaborate with this research group in the development of a software application for a Real-Time Flood Alert System. Our collaboration will consist of sharing information and working together in the specification of requirements, design and prototyping of the software system, and the use of the prototype for testing purposes. In the eventual success of this effort, we will consider its incorporation into our normal operations.

We believe that this development project will enhance the capabilities of our agency.

Sincerely,

Carlos H. Montanez
Emergency Information Center Administrator
December 31, 1998

To Whom It May Concern:

Subject: Proposal Entitled An Event-Rule Framework for Supporting Heterogeneous Distributed System

Please accept this recommendation letter in support of proposal entitled "An Event-Rule Framework for Supporting Heterogeneous Distributed Systems" by Dr. Javier A. Arroyo Figueroa, Dr. Néstor J. Rodríguez and Dr. José A. Borges. This proposal will be submitted to the NSF Next-Generation Program.

We will actively collaborate with this research group in the development of a software application for a Real-Time Flood Alert System. Our collaboration will consist of sharing information and working together in the specification of requirements, design and prototyping of the software system, and the use of the prototype for testing purposes. In the eventual success of this effort, we will consider its incorporation into our normal operations.

We believe that this development project will enhance the capabilities of our Agency.

Sincerely,

Luis A. Suárez Sánchez, P.E.
Eng. Supvr. Hydrologic Studies
Phone 289-3258
To Whom it May Concern:

Please accept this recommendation letter in support of the proposal entitled "An Event-Rule Framework for Supporting Heterogeneous Distributed Systems" by Dr. Javier A. Arroyo-Figueroa, Dr. Nestor J. Rodriguez and Dr. Jose A. Borges. This proposal will be submitted to the NSF Next-Generation Program.

We will actively collaborate with this research group in the development of a software application for a Real-Time Flood Alert System. Our collaboration will consist on sharing information and working together in the specification of requirements, design and prototyping of the software system, and the use of the prototype for testing purposes. In the eventual success of this effort, we will consider its incorporation into our normal operations.

We believe that this development project will enhance the capabilities of our agency.

Sincerely,

Eloy Colon
Hydrologist
To whom it may concern:

Please accept this recommendation letter in support of the proposal entitled “An Event-Rule Framework for Supporting Heterogeneous Distributed Systems” by Dr. Javier A. Arroyo-Figueroa, Dr. Néstor J. Rodríguez and Dr. José A. Borges. This proposal will be submitted to the NSF Next-Generation Program.

Our agency operates an ALERT (Automated Local Evaluation in Real Time) system, a network of 34 raingages. Most of these raingages are located at the headwaters of rivers and streams in the mountain divide. These sensors transmit through radio signal to three workstations that receive rainfall data in real time located in the State Civil Defense (SCD), the National Weather Service (NWS) and the U.S. Geological Survey (USGS).

We also have a cooperative agreement with the U.S. Geological Survey that operates a network of about 100 rain and river gages. This network transmits data in the field sites through satellite to the ground station at the USGS office. Our agency, as well as the NWS, accesses data by computer through phone lines.

Both systems complement each other and made possible to alert residents of potential flooding downstream. However, the correlation within rainfall and streams flows in real time in our island is not an easy task. Any research in this area is of fundamental interest to us in our purpose to save lives.

We will actively collaborate with this research group in the development of a software application for a Real-Time Flood Alert System. Our collaboration will consist on sharing information and working together in the specification of requirements, design and prototyping of the software system, and the use of the prototype for testing purposes. In the eventual success of this effort, we will consider its incorporation into our normal operations.

We believe that this development project will enhance the capabilities of our agency.

Sincerely,

Mariano Vargas Díaz
Hazard Mitigation Director

Apartado 5127, Pta. de Tierra Station San Juan, Puerto Rico 00906 Tel. 724-0124 FAX 725-4244