Long Running Application Models and Co-operating Monitors

Asit Dan and Francis Parr
IBM T. J. Watson Research Center
Hawthorne, NY 10532

Abstract: *Business processes representing related interactions within a business organization as well as e-commerce interactions across multiple independent autonomous organizations span days and weeks in time. The software which supports or automates the activities of a single business is best modeled as a set of long running applications which interact repeatedly with the outside world. The execution of these long running applications needs to be managed and monitored because of their business value. The monitoring framework must deal with all the different requirements on long running applications: (i) multiple interactions with each autonomous business partner driven by contractual agreements, (ii) execution of multiple application steps in response to an business event (request or response from an business entity, pre-scheduled action or time-out, etc.), (iii) interaction with a specific business role (internal or external) to perform some application steps, (iv) support for extended transactional semantics (e.g., compensation framework) to accommodate activities that are long running and/or across independent business entities, and (v) finally, development and execution framework for ease of programming of business applications. Legacy application monitors (e.g., Workflow, Extended transaction monitor, Conversation monitor) are designed to satisfy specific subset of these requirements. We argue that while proposed as well as legacy long running application monitors satisfy some of these objectives, no single monitor currently supports all of these objectives. This paper discusses a design approach which could lead to a unified monitor and also shows how in the absence of a unified monitor, long running business applications can be effectively modeled and built using multiple co-operating monitors.*

1. Introduction

The business activities are continuing to be automated with the rapid advancement of computer hardware and software development technologies. The business automation process can be broadly categorized into three phases. The first phase can be viewed as development of application programs that can capture some aspects of the independent and isolated business operations. Typically, such applications are run by a business employee to aid any

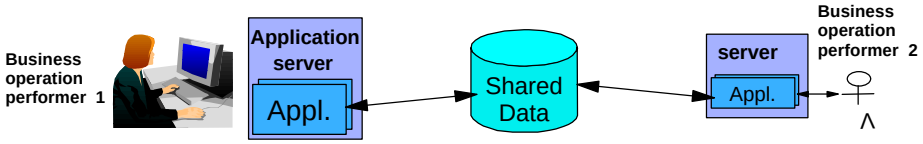

Fig. 1: Automation phase 1 *(Aiding individual business operations)*

of the numerous business operation steps (e.g., a decision making step, searching for a resource, buying or selling of a product, hiring/firing a new/existing employee, etc.). Each application is a short living code segment designed to perform a specific operation. A business employee upon interaction with a business partner (e.g., customer) or another business employee invokes a specific application, and provides appropriate input data. The application operates on this input data and/or command and creates/modifies/deletes/queries/derives some transient and/or persistent data. The advancement of TP monitors, database systems and other application servers mark this phase. The various persistent data elements (maintained in databases and/or file systems) as well as the knowledge

maintained by business employees represent the state of the business. These data are shared across independent applications (see Figure 1) in performing subsequent business operation steps.

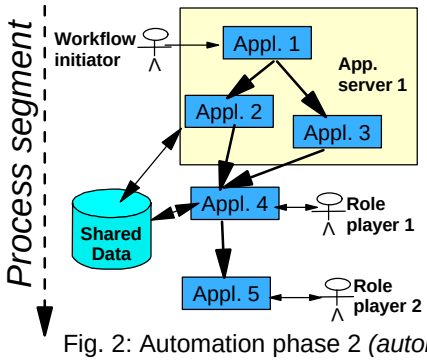

Fig. 2: Automation phase 2 *(automating a business process segment)*

The next phase of automation (see Figure 2) is underscored by the likes of technologies such as workflow, long running transactions, [1,2,3] etc., where multiple application steps are executed in succession, once it is initiated by a business activity. The data and/or state flows from one application step to another. Some of the application steps may be associated with specific business role players, while others may be executed without any interactions with business role players. The sequencing of application steps follow a pre-specified or dynamically modified control script representing a part of the business process. Various application server technologies (e.g., database, enterprise Java beans) enable easy sharing of persistent state across application steps and provide extended transactional and compensation semantics. Examples of such automated business processes are i) processing of an employee travel expense account where the employee creates and submits a travel expense reimbursement form, which is then automatically routed to the approver, followed by an actual reimbursement step performed by the finance dept. and ii) processing of a purchase order request which is first created by a sales representative after interaction with a customer, followed by a payment, packaging and shipping operations performed by appropriate business employees. While the automation of a business process segment facilitates significantly business operations, lack of a coherent view across multiple independent business process segments still creates a problem in managing interaction with the outside world. As an example, each time a customer interacts with a business organization, in the absence of such a coherent view, an independent internal business process segment without proper context may be triggered to satisfy this customer request. This is commonly referred to as the *Customer Relationship Management* (CRM) problem.

The network connectivity across business organizations introduces further automation of business operations captured via e-commerce applications. Businesses interact with one another according to some pre-agreed business interaction protocol [4,5] (e.g., OBI[6], SET). Related business interactions (e.g., sending a purchase order, modifying/cancelling a pre-submitted order, etc.) span over days and weeks. Such related interactions can be modeled as a long running conversation with simple interaction semantics. Each conversation is made of multiple actions, where each business partner plays a specific business role (e.g., buyer, seller, broker, etc.) supporting zero, one or more actions, and each action can be invoked by one or more conversation partners. Each action represents a request for an atomic service step, where the response for completion of a service request may be sent asynchronously. Additionally, each action may be cancelable, may have a responsiveness (time-out) requirement and its execution may need to follow an allowable sequence of invocations.

The state and history of each conversation are maintained by a long running interaction agent application at each business site [4,5] (see Figure 3). Upon receiving a request or response from another business, an *interaction agent* checks the validity of this interaction and enforces contractual agreements to protect this business site. If the interaction is deemed valid, it then invokes an internal business process segment as defined in automation phase 2. Note that an interaction agent may also invoke a business process segment in the absence of a desired interaction (e.g., time-out or lack of responsiveness, error handling upon receiving an undesirable request or response, etc.). The business process segments need not be modeled as independent, i.e., i) internal state data from one process segment may be used by the application steps in another process segment, and ii) a process segment may be aware of the overall business conversation. Lastly, multiple interaction agents may be used for a single business process segment,

and hence, the relationships across interaction agents needs to be maintained for assisting this business process segment (i.e., conversation correlation). For example, upon receiving a customer request via an interaction agent, a specific instance of a business process segment may send out procurement requests to its suppliers (e.g., travel agent sending requests to Hotel and Airlines) via other interaction agents.

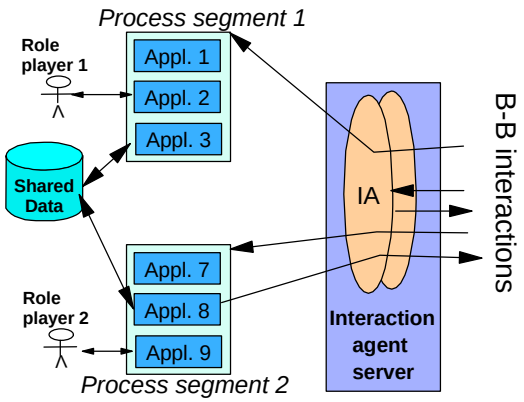

Fig. 3: Automation phase 3
(automating B-B interactions)

In summary, the activities of a business organization can be modeled as a set of long running applications. The business-to-business interactions can be modeled as long running conversations that follow contractual agreements, while internal business process segments as multi-step applications. The execution of these long running applications needs to be managed and monitored because of their business value. We therefore explore in this paper the details of various long running application models and the monitors that support them. The long running application models and monitors differ in terms of programming environment, mechanisms for sharing state and data across application steps and supported execution semantics, i.e., relationship across application steps. The overall monitoring framework however must deal with all the different requirements on long running applications: (i) multiple interactions with each autonomous business partner driven by contractual agreements, (ii) execution of multiple application steps in response to an business event (request or response from an business entity, pre-scheduled action or time-out, etc.), (iii) interaction with a specific business role (internal or external) to perform some application steps, (iv) support for extended transactional semantics (e.g., compensation framework) to accommodate activities that are long running and/or across independent business entities, and (v) finally, development and execution framework for ease of programming of business applications. We argue that while proposed as well as legacy long running application monitors satisfy some of these objectives, no single monitor currently supports all of these objectives. This paper discusses a design approach which could lead to a unified monitor and also shows how in the absence of a unified monitor, long running business applications can be effectively modeled and built using multiple co-operating monitors.

Due to space limitation we omit concrete illustrations of buyer and seller business applications that interact using the OBI protocol, and invoke approval process at the buyer site and fulfillment process (e.g., payment, shipping) at the seller site. We summarize the capabilities of various application monitors, however, omit mapping of the illustrated examples on these monitors. Finally, we conclude this paper with a proposal for a unified monitor.

2. Different long running application models and styles of monitor

In this section we list and comment on a variety of application models and their associated monitors pointing to the particular strengths of each. We start with classes of monitors which are well known and available as established middleware; subsequently we describe some more recent, emerging monitor technologies. After characterizing the monitor environments we discuss some of the situations in which combination of monitors can be used to allow applications tuned for different environments to interact, and the motivations for this.

- **Classical ACID transaction monitors e.g. CICS, IMS, Encina, Tuxedo:** The advantages of these have been discussed widely - a simple application programming model for concurrent server environment, data integrity, recovery, server throughput. The applications written to this programming model are not individually long running - most monitors in this class require transaction lifetime to be very short for good server throughput. But ACID transactions are the common form of legacy business applications and hence will be the basic steps in useful long running applications.
- **Collaboration Monitors e.g., Lotus Notes**: Here application are constructed with less emphasis on intensive throughput and access to protected data, but using powerful document oriented tools, for assembling documents, form handling, and scripting of more complex flows. Documents and document libraries provide the basis for application which save state over periods of time.
- **Object monitors as in Enterprise Java Servers, Corba Servers , IBM's Websphere**: These provide an object oriented programming model which maximizes software reuse and productivity in suitable skilled hands. This technology is also very effective for wrapping existing legacy applications and other functionality such as database. Persistent state objects enable longevity of long running applications.
- **Workflow monitors e.g., MQSeries Workflow, WfMC standard servers, etc**.: These monitors provide an environment for defining new processes as a control flow through sequences of 'black box' activities each of which may be executed either by launching a program or by triggering an interaction with a human operator. The workflow monitor takes responsibility for recovery of the control of the flow, after a failure of some activity or of the infrastructure and also schedules activities to available and appropriately skilled human operators using knowledge of the skills and roles of the pool of operators. In a workflow monitor , the flow interacts with activities only to start them , possibly passing in data and to complete them receiving data output. This clear separation between control flow and activities, is a strength in that it allows very disparate activities to be combined into a single process flow, but also a potential weakness in that if the implementation of the activity is a program which touches other recoverable resources such as a transactional database, recovery of the control flow may not be the same thing as fully recovering the state of the business process.

Moving to less well established , emerging technologies for long running application models and monitors, we have:

- **Service contract monitor**: In an Internet commerce environment it is important to be able to specify contracted behavior and conversational properties at an interaction point typically on an organizational boundary[4,5]. Furthermore, these boundary agreements must be specified without exposing the implementation used to support them which may be proprietary to the implementing organizations. A service contract monitor provides tool for creating, managing and even enforcing these trading partner agreements associated with long running application. They differ from say classical object interfaces in including requirements on patterns of conversational interactions, response time guarantees, etc.
- **Event Reaction and rule based Application monitors:** In this application structuring approach, rules are supplied to determine which pieces of application logic should be triggered by different combinations of events[2,5], - user interactions, completions, arriving requests, etc. The monitor uses these to control application processing. The way of structuring logic is particularly effective in a highly parallel environments where related activities may complete in different orders and complex conditions. Some event reaction monitors are essentially stateless - history is not required in order to correctly process each event as it arrives. Others have rules which depend on previous state of this conversation[4,5]. For long running applications in this class, the infrastructure must provide some effective automated scheme for retrieving the conversation state rather than leaving this to the application. A disadvantage of simple event reaction monitors is that as the number of rules becomes large it becomes increasingly difficult to understand the resulting flow through any intended business process. Combining procedural method of selecting a set of rules for immediate consideration with ERA can be effective for long running applications handling external events.
- **Long running UOW and Sagas**: Transaction models with less rigorous constraints than for strict ACID semantics have been proposed for long running business units of work (UOW) extending over days or weeks[7]. These models use a number of techniques including compensation steps, for user written reconciliation steps to resolve conflicts. They may also need optimistic or other advanced locking and automated versioning of data to minimize locking across application steps.

A highly desirable property for a business application environment supporting some user or trading partner, is that the long running application serving this partner has access and can react to the entire history of this particular partner's interactions. Good business interactions never end or forget previous interaction history. This remains true even when at different points in time this trading partner provides support with specific applications written in different styles to execute under different monitors. This prompts discussion of some of the ways that different ways in which monitors can interact.

2.1 Modes of Monitor Interaction

- **Launch of one monitor from another**. e.g., monitor A running application A1 launches application B1 running under monitor B. This is a very common mode of interaction meaningful for almost all monitor combinations. Note that some properties are typically lost in cross monitor launch. A workflow which launches a n ACID transactional activity and then acts to restart and recover the control of the workflow may find that the activity is not recovered (since states of active transactions at the time of failure are unknown.)
- **Layered : one or more monitors use shared underlying monitor for lower level interaction.** For example there may be a collaboration and a TPA monitor both using an underlying object monitor as the base for their runtime system.. Both the control flow and the activities of a workflow monitor could be implemented either as acid transactions with a single underlying monitor or object services within a single monitor. This sharing of concepts from an infrastructure allows much richer interactions - state sharing of conversations, process recovery etc. than is available in a simple launch interaction.
- **Structured: monitor relationships** Some combinations have naturally complementary properties and are therefore used to reflect this layering of the environment. For example, we have noted that sets of event reaction rules are highly effective at local synchronization, and handling external events, but become unwieldy when the number of rules becomes large. Hence partitioning event sets into workflow processes and having rules to manage event flows within a process is a useful combination. Similarly a TPA monitor may be needed to manage the interaction between autonomous organizations, while the implementation of applications within each organization may be handled using an application model which addresses program logic more directly.

3. Reflections on the feasibility of a "universal" monitor for long running applications

It is desirable to be able to get at the benefits of all these environments conveniently when needed. However, we know of no such monitor available today. Naively unioning all the features of all monitors mentioned above would lead to an application model of such complexity that no application would ever be written to it. (Frankinmonitor?) Therefore, it is highly desirable for an application designer to be able to start without pre-binding a specific choice of monitor - and have tooling which allows parts of the long running application to be deployed into the most appropriate monitor environment late in the development process. There are also runtime services, load balancing, remote operation which it would be highly desirable to have in common across sets of servers executing different monitors. Use of modern object technology as a lower layer substrate for other monitors can help provide uniform inter-monitor interactions. Consideration of interaction modes of monitors discussed above is a useful starting point for the design of broader ranging monitor environments

1. References

1. G. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, R. Gunthor and C. Mohan, ``Advanced Transaction Models in Workflow Contexts" In 12th *ICDE*, New Orleans, Lousiana, Feb. 1996.
2. U. Dayal, M. Hsu, and R. Ladin, ``Organizing Long-Running Activities with Triggers and Transactions" *ACM SIGMOD Record*, pp. 204-210, 1990.
3. **FlowMark**, SBOF-8427-00, IBM Corp., 1996.
4. A. Dan and F. N. Parr, "The Coyote Approach for Network Centric Business Service Applications: Conversational Service Transactions, a Monitor and an Application Style", HPTS workshop, Asilomar, CA, 1997.
5. A. Dan, et. al., ``The Coyote Project: Framework for Multi-party E-Commerce"**,** *seventh Delos workshop on e-commerce*, Crete
6. ***The Open Buying on the Internet (OBI) Consortium****, http://www.openbuy.org/*
7. *H. Garcia-Molina, and K. Salem, ``SAGAS,'' Proc. of ACM SIGMOD Conf., 1987, pp. 249--259.*

Long Running Application Models and Co-operating Monitors

Asit Dan and Francis Parr
IBM T. J. Watson Research Center
Hawthorne, NY 10532

Abstract: *Business processes representing related interactions within a business organization as well as e-commerce interactions across multiple independent autonomous organizations span days and weeks in time. The software which supports or automates the activities of a single business is best modeled as a set of long running applications which interact repeatedly with the outside world. The execution of these long running applications needs to be managed and monitored because of their business value. The monitoring framework must deal with all the different requirements on long running applications: (i) multiple interactions with each autonomous business partner driven by contractual agreements, (ii) execution of multiple application steps in response to an business event (request or response from an business entity, pre-scheduled action or time-out, etc.), (iii) interaction with a specific business role (internal or external) to perform some application steps, (iv) support for extended transactional semantics (e.g., compensation framework) to accommodate activities that are long running and/or across independent business entities, and (v) finally, development and execution framework for ease of programming of business applications. Legacy application monitors (e.g., Workflow, Extended transaction monitor, Conversation monitor) are designed to satisfy specific subset of these requirements. We argue that while proposed as well as legacy long running application monitors satisfy some of these objectives, no single monitor currently supports all of these objectives. This paper discusses a design approach which could lead to a unified monitor and also shows how in the absence of a unified monitor, long running business applications can be effectively modeled and built using multiple co-operating monitors.*

1. Introduction

The business activities are continuing to be automated with the rapid advancement of computer hardware and software development technologies. The business automation process can be broadly categorized into three phases. The first phase can be viewed as development of application programs that can capture some aspects of the independent and isolated business operations. Typically, such applications are run by a business employee to aid any

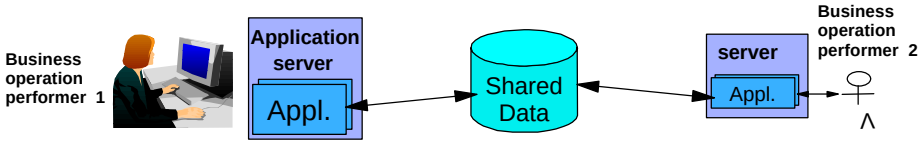

Fig. 1: Automation phase 1 *(Aiding individual business operations)*

of the numerous business operation steps (e.g., a decision making step, searching for a resource, buying or selling of a product, hiring/firing a new/existing employee, etc.). Each application is a short living code segment designed to perform a specific operation. A business employee upon interaction with a business partner (e.g., customer) or another business employee invokes a specific application, and provides appropriate input data. The application operates on this input data and/or command and creates/modifies/deletes/queries/derives some transient and/or persistent data. The advancement of TP monitors, database systems and other application servers mark this phase. The various persistent data elements (maintained in databases and/or file systems) as well as the knowledge

maintained by business employees represent the state of the business. These data are shared across independent applications (see Figure 1) in performing subsequent business operation steps.

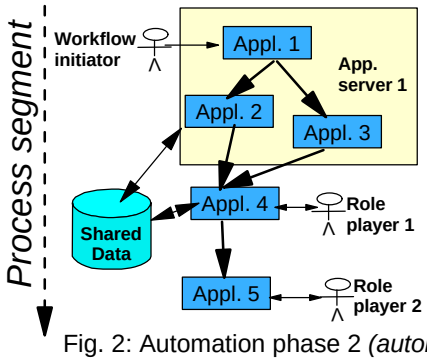

Fig. 2: Automation phase 2 *(automating a business process segment)*

The next phase of automation (see Figure 2) is underscored by the likes of technologies such as workflow, long running transactions, [1,2,3] etc., where multiple application steps are executed in succession, once it is initiated by a business activity. The data and/or state flows from one application step to another. Some of the application steps may be associated with specific business role players, while others may be executed without any interactions with business role players. The sequencing of application steps follow a pre-specified or dynamically modified control script representing a part of the business process. Various application server technologies (e.g., database, enterprise Java beans) enable easy sharing of persistent state across application steps and provide extended transactional and compensation semantics. Examples of such automated business processes are i) processing of an employee travel expense account where the employee creates and submits a travel expense reimbursement form, which is then automatically routed to the approver, followed by an actual reimbursement step performed by the finance dept. and ii) processing of a purchase order request which is first created by a sales representative after interaction with a customer, followed by a payment, packaging and shipping operations performed by appropriate business employees. While the automation of a business process segment facilitates significantly business operations, lack of a coherent view across multiple independent business process segments still creates a problem in managing interaction with the outside world. As an example, each time a customer interacts with a business organization, in the absence of such a coherent view, an independent internal business process segment without proper context may be triggered to satisfy this customer request. This is commonly referred to as the *Customer Relationship Management* (CRM) problem.

The network connectivity across business organizations introduces further automation of business operations captured via e-commerce applications. Businesses interact with one another according to some pre-agreed business interaction protocol [4,5] (e.g., OBI[6], SET). Related business interactions (e.g., sending a purchase order, modifying/cancelling a pre-submitted order, etc.) span over days and weeks. Such related interactions can be modeled as a long running conversation with simple interaction semantics. Each conversation is made of multiple actions, where each business partner plays a specific business role (e.g., buyer, seller, broker, etc.) supporting zero, one or more actions, and each action can be invoked by one or more conversation partners. Each action represents a request for an atomic service step, where the response for completion of a service request may be sent asynchronously. Additionally, each action may be cancelable, may have a responsiveness (time-out) requirement and its execution may need to follow an allowable sequence of invocations.

The state and history of each conversation are maintained by a long running interaction agent application at each business site [4,5] (see Figure 3). Upon receiving a request or response from another business, an *interaction agent* checks the validity of this interaction and enforces contractual agreements to protect this business site. If the interaction is deemed valid, it then invokes an internal business process segment as defined in automation phase 2. Note that an interaction agent may also invoke a business process segment in the absence of a desired interaction (e.g., time-out or lack of responsiveness, error handling upon receiving an undesirable request or response, etc.). The business process segments need not be modeled as independent, i.e., i) internal state data from one process segment may be used by the application steps in another process segment, and ii) a process segment may be aware of the overall business conversation. Lastly, multiple interaction agents may be used for a single business process segment,

and hence, the relationships across interaction agents needs to be maintained for assisting this business process segment (i.e., conversation correlation). For example, upon receiving a customer request via an interaction agent, a specific instance of a business process segment may send out procurement requests to its suppliers (e.g., travel agent sending requests to Hotel and Airlines) via other interaction agents.

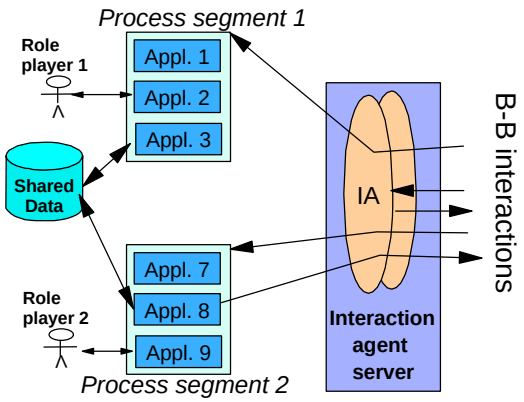

Fig. 3: Automation phase 3
(automating B-B interactions)

In summary, the activities of a business organization can be modeled as a set of long running applications. The business-to-business interactions can be modeled as long running conversations that follow contractual agreements, while internal business process segments as multi-step applications. The execution of these long running applications needs to be managed and monitored because of their business value. We therefore explore in this paper the details of various long running application models and the monitors that support them. The long running application models and monitors differ in terms of programming environment, mechanisms for sharing state and data across application steps and supported execution semantics, i.e., relationship across application steps. The overall monitoring framework however must deal with all the different requirements on long running applications: (i) multiple interactions with each autonomous business partner driven by contractual agreements, (ii) execution of multiple application steps in response to an business event (request or response from an business entity, pre-scheduled action or time-out, etc.), (iii) interaction with a specific business role (internal or external) to perform some application steps, (iv) support for extended transactional semantics (e.g., compensation framework) to accommodate activities that are long running and/or across independent business entities, and (v) finally, development and execution framework for ease of programming of business applications. We argue that while proposed as well as legacy long running application monitors satisfy some of these objectives, no single monitor currently supports all of these objectives. This paper discusses a design approach which could lead to a unified monitor and also shows how in the absence of a unified monitor, long running business applications can be effectively modeled and built using multiple co-operating monitors.

Due to space limitation we omit concrete illustrations of buyer and seller business applications that interact using the OBI protocol, and invoke approval process at the buyer site and fulfillment process (e.g., payment, shipping) at the seller site. We summarize the capabilities of various application monitors, however, omit mapping of the illustrated examples on these monitors. Finally, we conclude this paper with a proposal for a unified monitor.

2. Different long running application models and styles of monitor

In this section we list and comment on a variety of application models and their associated monitors pointing to the particular strengths of each. We start with classes of monitors which are well known and available as established middleware; subsequently we describe some more recent, emerging monitor technologies. After characterizing the monitor environments we discuss some of the situations in which combination of monitors can be used to allow applications tuned for different environments to interact, and the motivations for this.

- **Classical ACID transaction monitors e.g. CICS, IMS, Encina, Tuxedo:** The advantages of these have been discussed widely - a simple application programming model for concurrent server environment, data integrity, recovery, server throughput. The applications written to this programming model are not individually long running - most monitors in this class require transaction lifetime to be very short for good server throughput. But ACID transactions are the common form of legacy business applications and hence will be the basic steps in useful long running applications.
- **Collaboration Monitors e.g., Lotus Notes**: Here application are constructed with less emphasis on intensive throughput and access to protected data, but using powerful document oriented tools, for assembling documents, form handling, and scripting of more complex flows. Documents and document libraries provide the basis for application which save state over periods of time.
- **Object monitors as in Enterprise Java Servers, Corba Servers , IBM's Websphere**: These provide an object oriented programming model which maximizes software reuse and productivity in suitable skilled hands. This technology is also very effective for wrapping existing legacy applications and other functionality such as database. Persistent state objects enable longevity of long running applications.
- **Workflow monitors e.g., MQSeries Workflow, WfMC standard servers, etc**.: These monitors provide an environment for defining new processes as a control flow through sequences of 'black box' activities each of which may be executed either by launching a program or by triggering an interaction with a human operator. The workflow monitor takes responsibility for recovery of the control of the flow, after a failure of some activity or of the infrastructure and also schedules activities to available and appropriately skilled human operators using knowledge of the skills and roles of the pool of operators. In a workflow monitor , the flow interacts with activities only to start them , possibly passing in data and to complete them receiving data output. This clear separation between control flow and activities, is a strength in that it allows very disparate activities to be combined into a single process flow, but also a potential weakness in that if the implementation of the activity is a program which touches other recoverable resources such as a transactional database, recovery of the control flow may not be the same thing as fully recovering the state of the business process.

Moving to less well established , emerging technologies for long running application models and monitors, we have:

- **Service contract monitor**: In an Internet commerce environment it is important to be able to specify contracted behavior and conversational properties at an interaction point typically on an organizational boundary[4,5]. Furthermore, these boundary agreements must be specified without exposing the implementation used to support them which may be proprietary to the implementing organizations. A service contract monitor provides tool for creating, managing and even enforcing these trading partner agreements associated with long running application. They differ from say classical object interfaces in including requirements on patterns of conversational interactions, response time guarantees, etc.
- **Event Reaction and rule based Application monitors:** In this application structuring approach, rules are supplied to determine which pieces of application logic should be triggered by different combinations of events[2,5], - user interactions, completions, arriving requests, etc. The monitor uses these to control application processing. The way of structuring logic is particularly effective in a highly parallel environments where related activities may complete in different orders and complex conditions. Some event reaction monitors are essentially stateless - history is not required in order to correctly process each event as it arrives. Others have rules which depend on previous state of this conversation[4,5]. For long running applications in this class, the infrastructure must provide some effective automated scheme for retrieving the conversation state rather than leaving this to the application. A disadvantage of simple event reaction monitors is that as the number of rules becomes large it becomes increasingly difficult to understand the resulting flow through any intended business process. Combining procedural method of selecting a set of rules for immediate consideration with ERA can be effective for long running applications handling external events.
- **Long running UOW and Sagas**: Transaction models with less rigorous constraints than for strict ACID semantics have been proposed for long running business units of work (UOW) extending over days or weeks[7]. These models use a number of techniques including compensation steps, for user written reconciliation steps to resolve conflicts. They may also need optimistic or other advanced locking and automated versioning of data to minimize locking across application steps.

A highly desirable property for a business application environment supporting some user or trading partner, is that the long running application serving this partner has access and can react to the entire history of this particular partner's interactions. Good business interactions never end or forget previous interaction history. This remains true even when at different points in time this trading partner provides support with specific applications written in different styles to execute under different monitors. This prompts discussion of some of the ways that different ways in which monitors can interact.

2.1 Modes of Monitor Interaction

- **Launch of one monitor from another**. e.g., monitor A running application A1 launches application B1 running under monitor B. This is a very common mode of interaction meaningful for almost all monitor combinations. Note that some properties are typically lost in cross monitor launch. A workflow which launches a n ACID transactional activity and then acts to restart and recover the control of the workflow may find that the activity is not recovered (since states of active transactions at the time of failure are unknown.)
- **Layered : one or more monitors use shared underlying monitor for lower level interaction.** For example there may be a collaboration and a TPA monitor both using an underlying object monitor as the base for their runtime system.. Both the control flow and the activities of a workflow monitor could be implemented either as acid transactions with a single underlying monitor or object services within a single monitor. This sharing of concepts from an infrastructure allows much richer interactions - state sharing of conversations, process recovery etc. than is available in a simple launch interaction.
- **Structured: monitor relationships** Some combinations have naturally complementary properties and are therefore used to reflect this layering of the environment. For example, we have noted that sets of event reaction rules are highly effective at local synchronization, and handling external events, but become unwieldy when the number of rules becomes large. Hence partitioning event sets into workflow processes and having rules to manage event flows within a process is a useful combination. Similarly a TPA monitor may be needed to manage the interaction between autonomous organizations, while the implementation of applications within each organization may be handled using an application model which addresses program logic more directly.

3. Reflections on the feasibility of a "universal" monitor for long running applications

It is desirable to be able to get at the benefits of all these environments conveniently when needed. However, we know of no such monitor available today. Naively unioning all the features of all monitors mentioned above would lead to an application model of such complexity that no application would ever be written to it. (Frankinmonitor?) Therefore, it is highly desirable for an application designer to be able to start without pre-binding a specific choice of monitor - and have tooling which allows parts of the long running application to be deployed into the most appropriate monitor environment late in the development process. There are also runtime services, load balancing, remote operation which it would be highly desirable to have in common across sets of servers executing different monitors. Use of modern object technology as a lower layer substrate for other monitors can help provide uniform inter-monitor interactions. Consideration of interaction modes of monitors discussed above is a useful starting point for the design of broader ranging monitor environments

1. References

1. G. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, R. Gunthor and C. Mohan, ``Advanced Transaction Models in Workflow Contexts" In 12th *ICDE*, New Orleans, Lousiana, Feb. 1996.
2. U. Dayal, M. Hsu, and R. Ladin, ``Organizing Long-Running Activities with Triggers and Transactions" *ACM SIGMOD Record*, pp. 204-210, 1990.
3. **FlowMark**, SBOF-8427-00, IBM Corp., 1996.
4. A. Dan and F. N. Parr, "The Coyote Approach for Network Centric Business Service Applications: Conversational Service Transactions, a Monitor and an Application Style", HPTS workshop, Asilomar, CA, 1997.
5. A. Dan, et. al., ``The Coyote Project: Framework for Multi-party E-Commerce"**,** *seventh Delos workshop on e-commerce*, Crete
6. ***The Open Buying on the Internet (OBI) Consortium****, http://www.openbuy.org/*
7. *H. Garcia-Molina, and K. Salem, ``SAGAS,'' Proc. of ACM SIGMOD Conf., 1987, pp. 249--259.*