From Coordination of Workflow and Group Activities to Composition and Management of Virtual Enterprises

Marek Rusinkiewicz and Dimitrios Georgakopoulos
MCC, 3500 West Balcones Center Drive, Austin, Texas 78759
{marek, dimitris@mcc.com

Abstract
The objective of the Collaboration Management Infrastructure (CMI) project at MCC is the development of technologies that support a wide class of advanced applications requiring synchronous and asynchronous coordination of activities executed by human and software agents. CMI integrates a number of technologies having their origin in workflow and process management, groupware, and document management and provides additional solutions to the problems of dynamic process modification at run time, customizable awareness of the process status, and integration of external services. We describe the Collaboration Management Model and describe briefly its implementation using a collection of commercial tools augmented with new components providing the advanced functionality. We also describe briefly further extension to the model needed to provide support for a new class of applications implementing the concept of a virtual enterprise.

1. Introduction
In recent years we have witnessed significant progress in the development of work activity coordination technologies and their applications.

• Workflow Management has become a major industry and many classes of applications such as insurance claim processing or loan approval in banks are now largely automated. Workflow management products support coordination of asynchronously executed activities, dynamic assignment of individual tasks to roles within the process, data and control flow, and awareness of the status of the process.

• Process management techniques have been developed to support the various phases of software engineering, including the gathering and tracking of requirements, development of software modules, keeping track of software dependencies, and controlling the change.

• Groupware technologies providing support for Computer-Assisted Cooperative Work have matured and now provide support for distributed document management and distributed meetings, including sharing of information artifacts, and control of access to shared resources.

As a result, there are increased expectations that computer technology will help to fully automate applications that require coordinated interactions between human and software agents. Unfortunately, despite the tremendous progress, these expectations cannot be met in a number of important emerging applications that seem to pose new challenges for computer-assisted coordination technologies. Attempts to apply the existing technologies in advanced applications, such as crisis mitigation, support for military air and ground operations, or multi-enterprise service and process fusion, indicate that these applications present a set of new requirements which cannot be adequately addressed by the existing technologies. Below we will briefly enumerate these new requirements and suggest the possible approaches that may be used to address them.

1) Need for a rapid process prototyping, where new processes must be defined quickly, in response to requests. A possible approach is based on the concept of activity placeholders that allow plugging in existing process templates into process definition, or instantiation of process’ fragments at run-time.

2) Requirements to provide flexible process coordination with the actual execution path of a process determined at run-time. A possible approach is based on the concept of optional activities that might be included in the process definition, with the actual decision of whether an activity should be performed in the context of a particular process instance delayed until run-time.

3) Need to provide the capability to alter and expand the execution path through process escalation A possible approach is based on the concept of dynamic creation of task forces that might be assigned to specific tasks, introduction of situational roles (in addition to permanent organizational roles), and delegation of activities to specific users or roles.

4) Need to provide the process participants with access to shared info information workshops including the artifacts needed to perform an action and specific tools needed to manipulate them. A possible approach is based on the concept of context-specific resources.
5) Need to provide support for informed and timely decision making. A possible approach is based on the concept of time-varying situational awareness and windows of opportunity, for providing the information to the decision maker, so that s/he can act on it.

6) Need to support incorporation of external services into a process definition and enactment. A possible approach is based on the concept of service activities that may be included in the process definition and selected at runtime, in accordance with a pre-specified policy.

In this paper, we address these new requirements proposing preliminary solutions to some of them and outlining the directions of necessary research. In some cases the requirements can be addressed by combining technologies that were developed in different domains, while other requirements call for fundamentally new concepts and approaches.

The remainder of this paper is organized as follows. In the next section we will describe the project carried out at MCC with the intention of overcoming some of the limitations of existing technologies that present impediment to the application of computer assisted coordination techniques in advanced application domains. Section 3, will present our understanding of most important gaps in the current technology, which, when filled, would allow us to make a major progress in the creation of virtual enterprises (Virtual Enterprises constitute probably one of the most important new application domains for computer assisted cooperation technologies). In Section 4, we will review the related work.

2. CMI Project

The objective of the work on Collaboration Management Infrastructure (CMI) carried out at MCC is to provide: (1) advanced coordination of humans, applications, and services that may be provided by different enterprises, and (2) awareness, i.e., highly relevant information depending on the situation and role of each of participants. CMI technology development is driven by the requirements of many advanced applications that cannot be effectively supported by existing workflow and groupware technologies. In this paper, we describe CMI capabilities aimed at supporting crisis mitigation applications, military command and control, and service provisioning in a virtual telecommunications enterprise that are of interest to the companies supporting the CMI project. To address the requirements of such demanding applications, CMI provides a sophisticated Collaboration Management Model (CMM) and a corresponding component-oriented system that implements the CMM.

CMM utilizes and integrates existing primitives from workflow and groupware models and introduces new primitives that address previously unsupported requirements of the applications driving CMI. CMM is an extensible process model that consists of a Core Model (CORE) and several specialized extensions of it. CORE provides the basis for all extensions including a common set of process resources. In addition to traditional resources such as data, invoked applications, and organizational roles (that are discussed in the workflow literature [23, 17], CORE provides situational roles and context resources that provide key support for process escalation and awareness. These novel resource types are critical in supporting many other advanced applications.

Context resources serve two purposes. The first is to provide a shared workspace for communication between activities that may be distributed in a way that does not allow communication via normal dataflow. For example, consider two groups of people participating in different (sub)processes that need to exchange progress reports. A common context resource can be provided to the activities of these task forces to store references to their progress reports. The second important use of the context resource is to store and provide operations to manipulate situational roles. Such advanced roles can be can be dynamically created and may exist only within a context. Since context resources may have either pass-by-value or pass-by-reference scopes, situational roles are visible only to those activity instances that have access to the enclosing context resource. Therefore, each situational role has a scope that is determined by the visibility of the context resource that stores it, unlike organizational roles that have only global scope. Situational roles can be used within a process specification in the same way as usual global roles. For example, building a task force involves selecting a task force leader and task force members. The task force leader and member roles must be dynamically created and assigned to the specific individuals that have been selected to participate in each particular task force. In general, these roles are independent from the organizational roles of these people. They are only valid inside the task force and their lifetimes are restricted to that of the task force.

The CMM extensions are designed to support coordination and awareness. The Coordination Model (CM) provides standard data and control flow primitives similar to those that are currently supported by Workflow Management Systems (WiMS). In addition, CM provides advanced primitives for specifying process templates and escalating them [12]. CMM primitives for process escalation support rapid process prototyping, flexible coordination, and controlled dynamic process extension or refinement. For example, they enable extending a process template instance by adding new process templates, creating new roles and task forces as needed to deal with the current demands, or delegating work. Escalation is dynamically performed by coordinators and experts that may be humans or applications. Coordinators are responsible for determining the course of the process escalation and delegating responsibilities to process participants and task forces. Experts perform special-
ized activities in the process. CM allows explicit control of possible escalation points to avoid chaotic escalation and inefficiencies.

The Awareness Model (AM) is a CORE extension that captures awareness. We define awareness as information that is highly relevant to a specific role and situation of a process participant. Because a human's attention is a finite resource that must be optimized, awareness information must be digested into a useful form and delivered to exactly the users who need it. If given too little or improperly targeted information, users that make decisions will act inappropriately or be less effective. With too much information, decision makers must deal with an information overload that adds to their work and masks important information. AM allows customizing awareness via awareness specifications. Awareness specifications, which are provided by process/awareness designers, define what information should be directed to what users based on their roles in the process.

The Service Model (SM) allows the integration of external services in a CMI process. External services are captured by service interfaces that model external services as state machines. Such machines may include application-specific states, and operations can cause state changes. SM also provides service activities that are service proxies and can be used as “normal” activities in CMI process models. Service activities perform the actual service invocation and feedback operations defined in the service interfaces. SM deal with the autonomy and the heterogeneity of services by supporting service wrapper processes and removing many of the limitations of existing proxy mechanisms.

Further extensions can be introduced to support process evaluation and application-specific process models [12]. Figure 1 indicates this by showing application-specific extensions on top of the CM, SM, and AM.

![Figure 1. The CMM CORE + Extensions](image)

To allow application modeling, CMM provides types for activities, activity states, resources, and dependencies. The CMM types are used to develop application models. These models are instantiated during enactment. The basic primitives of the CMM are activity, activity state, resource, and dependency types. Process activity types consist of an activity state variable, activity variables, resource variables, and dependency variables. Activity variables represent the sub-activities of a process. Resource variables describe the resources needed during process execution. Dependency variables define the coordination rules for the subactivities of the process, e.g., their order of execution. Basic activity types are restricted to an activity state variable and resource variables. It should be noted that the CMI activity and resource variables are generalizations of the standard activity and resource primitives in the Workflow Management Coalition (WfMC) reference model [36] and in many commercial WfMSs. In addition to default activity, state, resource, and dependency types, CMM provides meta-types allowing the creation of application-specific activity, state, and resource types.

2.1. CMI primitives for coordination

The CMI Coordination Model combines standard workflow primitives with advanced coordination primitives. The standard control and data flow primitives CMM provides, have been adopted from WfMSs and the Workflow Management Coalition (WfMC) standards. They have been extended, however, to adequately support dynamic applications, such as crisis mitigation.

To illustrate crisis mitigation requirements and motivate the advanced coordination primitives CMI provides consider a crisis as it may happen in an epidemic outbreak. Suppose an epidemic outbreak is reported in some region of the United States. The health organization for that region would start a health crisis mitigation process to determine the nature of the disease and contain the outbreak. Hence, a health crisis mitigation process typically consists of an information gathering process to determine the cause of the health crisis, and a containment process that deals with controlling the spreading of the epidemic and coordinates appropriate containment. These processes involves doctors and patients in the area of the outbreak, the Center for Disease Control or the World Health Organization, agencies involved in containing the outbreak, and news agencies. Each process may be coordinated by directors of the agencies involved. However, various experts that participate in the process, such as epidemiologists and microbiologists performing lab tests, may also influence the course of the evolving process.

While the details of the process are specific to the particular outbreak, a health crisis mitigation process involves practiced responses that are dynamically tailored for the situation. The crisis mitigation process starts when a health agency becomes aware of the outbreak through normal reporting channels. The process ends when the nature of the pathogen is understood. Depending on the specific crisis situation, coordinators determine activities that need to be performed and delegate these activities to individuals (i.e., experts or other coordinators) or task forces to perform them.

To support crisis mitigation and command control, CMM uses the definition and enactment of process templates. A
process template is a process specification that leaves certain aspects of the actual process instances open to be concretized at run-time. Process templates are created by process designers and they are escalated by coordinators and experts that participate in a process template instance. CMM allows the escalation of process templates during enactment. In particular, CMM-supported process template escalation allows rapid process prototyping, flexible coordination, and controlled dynamic extension or refinement. These enable coordinators and experts to introduce and perform activities that mitigate the crisis.

CMM allows process participants to select the specific type of activities to perform and exercise options. For example, in the health crisis process, an epidemiologist is an expert process participant that has the necessary expertise to determine the specific type of lab test to perform in the current situation of an epidemic. This can be accomplished by assignment of expert process participants (e.g., epidemiologists) to placeholder activities that may have repeated optional dependencies. Placeholders represent "generalized" activities (e.g., indicating the need or opportunity to perform a lab test, without prescribing the specific type of test to perform), and specify the point in the process where the participants assigned to them can decide the actual activity type to perform. An expert assigned to such a placeholder activity may select and instantiate the specific activity to perform from a list of existing activity types that match the signature (and possibly additional conditions) of the activity placeholder, or alternatively create and select a new activity type. Repeated optional dependencies capture optional activities that experts can perform at any time after they become enabled via normal control flow. Experts may perform such activities more than one time or choose not to perform them.

In addition, CMM permits controlled dynamic extension and refinement of processes by delegation of activities to other participants through dynamic role assignment to individuals or task forces. Situational role creation and assignment are key requirements for delegation of activities that require unanticipated skills and responsibilities. Task force participants typically have both organizational and situational task force-specific roles. Organizational roles correspond to the organizations the process participants belong to. Task force-specific roles are meaningful only in the situation of a specific task force. Task force-specific roles are dynamically created when the task force is created, assigned to task force members as they join the task force, and deleted when the task force is dissolved. Dynamic creation of task forces and task force-specific roles is needed in crisis mitigation to allow coordinators to dynamically and spontaneously deal with the current situation. For example, consider a task force for coordinating central labs with local investigators. The task force leader role may be assigned to an epidemiologist and is valid (has a scope) bound by the duration of the task force.

CMM allows rapid process prototyping, by allowing coordinators to dynamically extend a process by adding new process templates and instantiating them as subprocesses of the current process instance. CMM permits late binding of process templates to placeholder activities. Just like an expert can select an activity type to replace a placeholder, a coordinator can select a process template from a list of existing templates (or construct a new template if the existing templates are not appropriate). For example, in one health crisis a coordinator may replace a placeholder “identify cause” by selecting a template for identifying a pathogen, while in another crisis the situation may require a process template for identifying a toxin. Options allow the flexibility coordinators need as coordination experts.

The advanced CM coordination primitives for rapid process prototyping, flexible coordination, and controlled dynamic extension and refinement are discussed in more detail in the following sections. Placeholders are described in Section 2.1.1. Repeated optional dependencies are used to model windows of opportunity and they are introduced in Section 2.1.2. Primitive operations for activity delegation are discussed in Section 2.1.3. Advanced role assignment is presented in Section 2.1.4.

2.1.1. Activity placeholders. Activity placeholders allow for activities whose concrete types are unknown or intentionally left open at specification time of a process template. In CMM, an activity placeholder may be declared at any point in a process specification where a concrete activity can be declared. At runtime, activity placeholders can be replaced by activities with concrete activity types.

Activity placeholders consists of two parts: a placeholder activity variable and an (activity type) resolution policy. A placeholder activity variable is like any other activity variable in a process type. At runtime, the resolution policy determines the specific activity type to be used to replace the placeholder activity variable. The resolution policy ensures syntactic and semantic compatibility between the placeholder and the activity type that eventually replaces it. To ensure syntactic compatibility, the resolution policy limits the choice of process types that can replace a placeholder to those that have the same dataflow signature (i.e., input and output parameters) as the placeholder activity variable. Semantic compatibility with the application is provided by activity type selection or construction. Selection policies may range from making a simple choice to a full-blown process. More specifically, a simple selection policy may prompt an expert or a coordinator to pick an activity type from a list of suggested types or to browse/query an activity type repository for a suitable activity type. Selection policies in this category may utilize user profiles. A simple selection policy
can also be a script or a program that is automatically executed when a process participant instantiates a placeholder activity variable. A selection policy becomes a full-blown process in situations that require multiple participants to decide the actual activity type for the placeholder. For example, an epidemiologist may want a second opinion to decide which type of test (i.e., activity) to choose. In this case, a selection policy process for soliciting second opinions for lab tests may be specified as the selection policy.

Figure 2 depicts a simple example of an activity placeholder within a process type $P$.

![Figure 2. Example of an activity placeholder](image)

Suppose that the placeholder activity variable $av$ has an epidemiologist role $R$ assigned to it, and a resolution policy that allows the participant selected from $R$ at runtime to pick the actual activity type for the placeholder from a list $\{A_1,\ldots,A_n\}$. To ensure the semantic compatibility of the activity type that can be used to replace the placeholder in this application, the set of activity types from which the selection policy chooses one is explicitly provided in the placeholder selection policy.

### 2.1.2. Repeated optional dependencies and windows of opportunity

Advanced collaborative applications, particularly for crisis mitigation, are characterized by the fact that the objectives of these applications cannot be met by simply running an algorithm, e.g., in form of a predefined process. Instead, the potential means to achieve the application’s objective are often known, but usually the timing and frequency of their usage cannot be predetermined. In a process management system such as CMI the means for achieving application objectives are activities that may be invoked within a processes. However both the exact invocation place in the control flow path of the process and the number of invocations cannot be specified beforehand. The decision when and how often these activities are performed can only be made by the participants involved in a particular process instance. However, the opportunity to perform such optional activities may be limited to a specific window that opens and closes by the execution of specific activities in a process. In CMM windows of opportunity are provided by repeated optional dependencies. In addition, these dependencies can be used to capture unstructured activities, such as those used in groupware.

The repeated optional dependency primitive consists of a repeated option creator and a terminator. The repeated option creator specifies that when an activity variable, say $av_{opt}$, is enabled by normal control flow, it may be instantiated zero or more times. The time and number of instantiations is determined by the participant(s) assigned to $av_{opt}$. The repeated option terminator limits the time span within which $av_{opt}$ can be instantiated by the starting of another activity, say $av_{term}$. That is, as soon as $av_{term}$ starts no additional instances of $av_{opt}$ can be started. The process designer may specify an upper bound on the total number of instances of $av_{opt}$ that can be performed. Activity variables constrained by repeated optional dependencies cannot have outgoing dependencies to the rest of the activities in the process.

Figure 3 shows an example of the repeated optional dependency. The left hand side of the figure depicts a process type $P$ containing the activity variable $av$ which has a repeated optional dependency attached. If an instance of $P$ is executed, the activity $s$ has to be performed first, because it has no incoming control flow transitions. After $s$ ends, the control flow transitions originating at $s$ will evaluate to true, since there are no transition conditions in this example. This causes the enabling of the activity variable $av$. At this point $av$ is offered to the players of role $R$ for execution. Therefore, the completion of $s$ and the enabling of $av$ open the window of opportunity to perform $av$. The right hand side of Figure 3 assumes that participant $p_2$ chooses to perform an activity instance of $av$. This causes the creation of an activity instance $a_1$. In addition, $a_1$ is assigned to $p_2$ and gets the input resources as specified for $av$. The activity variable $av$ itself stays on the worklists of all players of role $R$. The remainder of the process (indicated as dots in Figure 3) proceeds independently of $av$, since $av$ is not permitted to have outgoing dependencies. If activity $e$ is performed the terminator of the repeated optional dependency will be triggered, disabling $av$, and $av$ will disappear from the worklists of those that play role $R$. Therefore, $e$ closes the window of opportunity to perform $av$. In the interval between ending of activity $s$ and starting activity $e$, the players of role $R$ can freely choose to perform activity instances belonging to $av$, or none at all.

In a crisis mitigation process the repeated optional dependency primitive provides for the specification of a list of suggested options that participants can perform in a certain point of the process. Suppose that the process in Figure 3 describes the work of an microbiologist in a hospital task force, say to identify a virus. Activity $s$ is an activity which supplies the microbiologist with information about the current situation in the progress of the epidemic. Suppose activity $av$ is a lab test that identifies the virus. After $s$, the microbiologist has many options on how to proceed. He/she may choose not to perform lab test $av$, or perform it several times. However, as soon as the task force leader calls for wrapping up the results, i.e., activity $e$ is performed, no additional lab tests may be started.
2.1.3. Primitive operations for activity delegation.

Coordinators escalate a process instance by: (1) creating new activities, as we discussed in Sections 2.1.1 and 2.1.2, and (2) delegating activities to other process participants. Delegation is a collective term for a number of coordination acts. In a crisis mitigation setting delegation ranges from the simple assignment of a participant (the delegatee) to an activity to assigning a complicated problem to a dynamically created task force where individual tasks force members play separate roles in resolving the problem. Complex delegation acts, such as a delegation to a task force, cannot be supported by a single primitive but require a delegation process that constitutes a combination of primitives.

To support simple delegation CMM adopts delegation primitives used by WfMS. A coordinator may delegate activities that have been originally assigned to her/him to another participant or role by performing a delegate operation on the activity to be delegated. A delegate activity operation is provided, for example, by the commercial WfMS COSA [27] and by the academic Mobile WfMS [23]. Another rudimentary primitive to support simple delegation settings is to determine the participant(s) and/or role(s) responsible for performing an activity by a preceding activity. Such a functionality is provided, for example, by IBM FlowMark [21]. CMM supports both of these simple variants of operations for activity delegation.

To support a specific variation of delegation to a group, such as a task force, CMM provides that advanced role assignment primitive which is described in the next section.

2.1.4. Advanced role assignment. In traditional WfMSs, an activity variable in a process specification results in exactly one activity instance that is assigned to only one of the process participants in the role associated with av. This traditional one-out-of-n role assignment is suitable for applications where work has to be distributed among a group of workers. However, more cooperative settings, like crisis mitigation processes, involve group activities that must be performed by multiple members of a group. Therefore, the activity variable of such activities results in multiple instances at runtime, and each instance must be assigned to a different group member.

CMM’s advanced assignment dependencies introduce an extension of the basic participant assignment by providing also m-out-of-n (m ≤ n) role assignment. The special case m = n is called an all assignment. An m-out-of-n (m ≤ n) assignment dependency that assigns a set of participants to an activity variable av leads to m activity instances, each of them performed by a different individual. Activity variables succeeding av in the process specification become ready only if all m activity instances corresponding to av have ended. The output resources produced by av’s activity instances can be combined into a single output of av using a data aggregation policy. If no aggregation policy is given, the last write to an output resource of av determines the value of this output resource.

CMM primitives can be combined to model complex processes. In particular, all primitives of the CORE can be combined with all CM primitives. Activity placeholders can be combined with either advanced assignment dependencies or repeated optional dependencies. Whenever such a combination occurs, the repeated option dependency is evaluated before the resolution policy for each activity instance. Activity variables with attached repeated optional dependencies cannot have outgoing control and dataflow dependencies, because the point in the process where the instantiations of such an activity variable terminate, is left to the participants to decide. Therefore, outgoing dependencies would never get enabled. However, such activities can exchange resources by using a shared context resource.
decisions and to react properly when unanticipated circumstances occur.

The Awareness Model (AM) is an extension to CORE that can provide such timely, highly relevant information to decision makers. Information in AM is specified and delivered as awareness events. Such events include activity state changes and resource status events. Furthermore, AM allows for the addition of new event types when new resource types are defined. Awareness events can be combined into composite awareness events. Delivery of detected composite events can be directed to users in either global or situational roles.

In AM, an event carries a set of name-value pairs called parameters that give detailed information about what has occurred. Because events are assumed to be self-contained, an event’s parameters completely describe the event (e.g., include type, time, and source). A composite event is an event that is defined to occur as a result of some non-empty collection of events called its constituent events. Because events are self-contained, composite events summarize the parameters of the constituent events. Composite events may be constituents of other composite events. Non-composite events, called primitive events, come from well-defined event producers. AM provides an awareness specification language that is used by process/awareness designers to construct awareness schemas. These define composite events, describe how constituent events are digested, and to whom the result is to be delivered.

Formally, an awareness schema on process schema P is defined to be an awareness description and an awareness delivery role. The awareness delivery role is an organizational or situational role known to process P that determines the set of users who may receive the information specified in the awareness description. The awareness description is a composite event specification that has been specialized for the processing of process enactment events a process type P. A composite event specification is a rooted, directed acyclic graph (DAG) where the leaves of the DAG are primitive event producers, the non-leaves are event operator instances, and the edges are connections, i.e., event streams, between event producers and the consuming slots of event operator instances. An event operator is a self-contained, reusable algorithm for recognizing instances of a pattern of constituent events and calculating the parameters of the resulting composite events. An event operator instance is a running instance of the operator’s algorithm which acts as a consumer of multiple typed event streams, called inputs, and produces a stream of events called the output. An event operator instance can be thought of as a computational pipeline that can produce any number of output events for a single input event.

During execution of the specification, primitive events enter the DAG at their associated leaves and flow to the input slots of operators connected to those leaves. As composite events are generated, they flow to their consumers, which are usually slots of other event operator instances. Composite events that are output from the root of the DAG are said to be composite events detected by the composite event specification. The entire composite event specification is an event producer for events produced by its root operator instance.

Awareness descriptions are based on an event processing model specialized for a process management system. Furthermore, AM event operators can be customized to capture application requirements by allowing specification of parameters (not to be confused with event parameters). The parameters, which are specified at design-time, allow process designers to customize the behavior of each operator instance. Thus, awareness descriptions are DAGs with parameterized operator instances. AM allows only a fixed palette of nine operators that can be classified as filtering, generic, count, comparison, and process invocation event operators. A more comprehensive discussion of CMI Awareness Model can be found in [4].

2.3. CMI primitives for integration of external services

The CMM Service Model (SM) allows the integration of external services in a CMI process. Utilizing services often involves conversations between the requester of a service (service client), and the service provider. For example, conversations between service clients and providers may involve the execution of a service request operation, various control and information exchange operations during the ongoing service performance, and the final delivery of the service results. Capturing and coordinating such conversations requires modeling and execution primitives that are far beyond the usual invocation/termination semantics of activities traditional workflow models. The situation is further complicated by possible heterogeneity between the service requester and the service provider. For example, there may be different protocols assumed by the service client and the service provider. Integrating heterogeneous services in a traditional workflow model typically results in specification explosion, combinatorial explosion of process activities and corresponding dependencies between them that is practically unmanageable.

To illustrate why such service-related requirements that cannot be effectively supported by existing process management technologies, we discuss a multi-enterprise process (MEP) for universal telecommunications service provisioning in a virtual telecommunications enterprise. We intend to illustrate that CMM addresses many key requirements for MEP management, including those described in our example MEP.
A multi-enterprise process (MEP) is a process that fuses together services provided by different enterprises. Such service typically encapsulates single-enterprise processes (SEPs). The SEPs encapsulated by each specific service S are the actual business processes used by the service provider of S to provide S. As an example of such a MEP, consider an Universal Telecommunications Service Provisioning process that allows clients to request local, long distance, wireless, and internet service from a virtual telecommunications enterprise (referred to as the universal service provider). A example Universal Telecommunications Service Provisioning process is depicted in Figure 4. Its top-level process starts when a customer of the universal service provider requests a Universal Telecommunications Service. Activity \( T_a \) involves an operator collecting information from the customer, or a customer directly providing the information via a web browser. When sufficient customer data are collected, activity \( T_b \) is performed to (1) verify that the information provided by the customer is accurate, and (2) create a corresponding universal service order record.

![Figure 4. Multi-enterprise Process for Universal Telecommunications Service Provisioning](image)

When \( T_b \) is completed, the top process starts a combination of the Provide Local Exchange Service, Provide Long Distance Service, Provide Wireless Service, and Provide Internet Service. These services are managed by different autonomous enterprises. When all selected subprocesses are completed, activity \( T_c \) is preformed to create a single bill for all telecommunications services that comprise the universal service. Finally, activity \( T_d \) involves a human operator who calls the customer to inform him/her of the establishment of the requested universal service and to verify that the provided service meets the customer needs.

The Local Exchange Service in Figure 4, is implemented by three SEPs that are encapsulated in the service. In particular, a SEP is used to provision and fulfill the local service, while other SEPs provide the service order status, and cancel the service request. From the perspective of service integration by a MEP, the SEPs for Long Distance, Wireless, and Internet Service Provisioning are similar to the Local Exchange Service Provisioning process. Therefore, we do not discuss them in further detail.

CMM supports services that are functional abstractions of a collection of SEPs provided by a single enterprise. In particular, CMM services can capture the following:

- **Control of Service** (CoS): Operations that start a SEP, provide input to the SEP, or synchronize the SEP with an activity of a client SEP, or cause the SEP to take a specific execution route.
- **Quality of Service** (QoS): Qualitative and quantitative measures of the performance of the SEP(s) in the service, such as time and cost, and combination of these measures.
- **Awareness of Service** (AoS): Process events caused by specific activity state changes in a specific SEP and comprehensive abstractions via composition of such events.

The functional interfaces for CoS, QoS, and AoS are specifically provided via explicit CMM service contracts. Legal contracts between the service provider and its clients are also part of a service contract, but they are outside the scope of this paper. CMM makes no particular assumptions about the implementation of the SEPs in a CMM service. Unlike many existing approaches that assume that SEPs in a service are implemented by a process management engine, CMM allows a variety SEP implementations that include CORBA servers, basic programs, or even legacy information systems.

CMM assumes that service designers require (1) no direct knowledge or access of the service models and implementations used by other enterprises, and (2) services that belong to different enterprises are integrated and interact only via a MEP. For example, consider the interaction between the Provide Local Exchange Service and Provide Long Distance Service in Figure 4. This dependency exists because the Long Distance Service cannot be activated unless the Local Exchange Service receives necessary information from the Long Distance Service, and vice versa. In particular, the Local Exchange Service must wait until the Long Distance Service produces the data required by the switch to activate long distance (e.g., the identification code of the long distance service provider). In addition, the activity in the Long Distance Service that generates the long distance service bill for the universal service must wait until the installation of the local exchange service is completed. These processes must be coordinated by the Universal Telecommunications Service MEP to ensure this behavior.

A unique contribution of CMM is the integration of the traditional notions of activity and service into a single CMM primitive, we refer to as service activity. Services are often conversational, i.e., they allow or require interaction where clients perform multiple service invocations and re-
cause state changes. SM may include application-specific states and operations that model external services as state machines that interfaces existing process and service models by providing to a specific generic state. CMM addresses the limitation of states and operations so that they can be always generalized to a specific state. CMM provide subclassing of application-specific activity that cause transitions between such service states. Finally, SM provide service wrapper process that hide service heterogeneity.

2.4. Implementation of CMI

The CMI system takes advantage of existing technology, by using IBM FlowMark version 2.3 [21] and MCC’s CEDMOS event processing system [6]. In this section we review the runtime architecture of the CMI system and describe how the CMM coordination and awareness primitives are implemented atop of COTS technology.

2.4.1. The runtime architecture of the CMI System.

Users of the CMI system are separated into two principal (not necessarily disjoint) classes: participants and designers. A participant is a human or program (machine) individual involved in a collaboration process. A designer is usually a person who creates and maintains CMM specifications. Participants interact with the CMI enactment system using CMI client tools for participants. These include a worklist tool, which contains activities that the participant is eligible to perform, a process monitoring tool, an awareness information viewer, and a hybrid tool that combines a worklist with some monitoring capabilities. CMI client tools for designers include tools for creating and maintaining process specifications for coordination, awareness, etc. Thus, the CMI client tools subsume the tools defined by the WfMC [17]. All client tools connect to the CMI enactment system (Figure 5).

Figure 5. CMI System Run-time Architecture

The CMI enactment system contains several engines that implement the CMM. The CORE Engine implements the primitives of the CORE. Therefore, it is used by the other engines that are responsible for the CM, AM, and other extensions of the CORE (e.g., the Service Model). The Coordination and Awareness Engines implement the CM and AM, respectively. In the following section we outline how CMI engines capitalize on the use of COTS technologies. The coordination and resource handling capabilities of a COTS WfMS, are exploited by the Coordination and CORE engines.
Engines. The Awareness Engine makes use of the CEDMOS event processing system.

2.4.2. Implementation of CMM primitives. The CMM provides basic and process activities. CMM basic activities are consistent with the WfMC standard [36] and are supported by virtually all commercial WfMS. Thus, each basic CMM activity uses an activity of the underlying WfMS. Implementation of CMM processes is not trivial, since it depends on the dependencies used in the process definition. Typically, a CMM process produces multiple WfMS processes that are orchestrated by the CMI Coordination Engine. The CMM helper resources are implemented by external applications of the underlying WfMS. The CMI system also uses the WfMS to maintain workflow internal data, traditional control and dataflow primitives, as well as one-out-of-n role assignment.

CMM context resources, in particular situational roles, and advanced coordination primitives for activity templates are beyond the functionality provided by commercial WfMS. The implementation of the advanced CMM primitives maps them into fragments of WfMS processes and/or new top-level WfMS processes. The translation process involves the introduction of hidden activities controlled by the CMM Coordination Engine. The hidden activities allow for the CMM Coordination Engine to influence the course of a WfMS process execution. As a consequence, an escalated CMM process template instance corresponds to a set of processes in the underlying WfMS. The CMM Coordination Engine maps, integrates, and orchestrates these WfMS processes. A more detailed discussion of the implementation of the CORE and Coordination Engines can be found in [13].

The awareness model (AM) implementation heavily leverages features provided by CEDMOS (Composite Event Detection and Monitoring System) [6], which directly supports the AM event processing model described in above. A process designer creates and modifies awareness specifications using the awareness specification tool, a GUI editor for authoring awareness schemas. At the end of an editing session, the tool emits code that captures the awareness specification(s) in the form of a composite event detector. The compiled code becomes part of the awareness engine. During execution, primitive events flow from the various CMM engines to the awareness engine. Detected composite events, called awareness events, have their delivery instructions evaluated to a list of users and the event’s information is queued for each user on the list. The awareness information viewer, a component of the CMI client for participants, is responsible for monitoring the awareness event queue for its user and displaying awareness events to him/her. The viewer enables a number of capabilities from a delivered event including viewing the details of the event, monitoring the process that generated the event, and viewing the awareness specification that generated the event. [4] discusses the AM implementation in more detail.

3. Future research directions

The process definition monitoring and enactment framework described in the previous section has been applied in applications ranging from medical crisis management and synchronization of military operations, to service provisioning in telecommunication industry. However, before the framework can be applied as a basis for electronic commerce applications based on the notion of Virtual Enterprise (VE), a number of problems related to service integration, conversation and feedback in real-time, would need to be solved. The most important problems include:

• service selection (ontologies to capture client needs and service capabilities)
• service composition and coordination (real-time mega programming of service integration using CMI’s process-oriented framework)
• service optimization (advertisement, local optimization/matchmaking between requirement and individual services, and global optimization/bulk and value chain optimization)
• service contracts (real time completion and validation of legal contracts)
• service ratings and policing (rating and probing services to validate advertised capabilities)

The most important problem is the brokering and selection from services that may be offered by multiple service providers. To maintain market agility, a Virtual Enterprise may use the services (and corresponding service providers) that offer the most favorable terms in achieving its business objectives. To select the best collection of services that achieve its business objectives, the service integrator may perform dynamic service selection and integration. For example, the Universal Service Provisioning may dynamically select the actual service providers, integrate their services as the Universal Service Provider, and use them to provide each component service whenever it is needed. The selection of each component service may depend on the Quality of Service they provide, as well as the awareness, control and the compatibility with the integrator. There are preliminary research contributions in the area of service quality and automatic selection via service brokering, e.g., [14].

From an integration perspective, a service interface specification—including application-specific activity operations, states, a corresponding state transition diagram, and input/output resources—captures these parts of the service semantics that are required for the service to be used within a process. However, the service interface is not suitable for initial service discovery at process definition time or for the discovery of newly offered services, since it requires that the services are already modeled. Therefore, additional
means to advertise and discover services in a VE environment are required that support the service life-cycle that covers the time before a service is captured and wrapped by a basic service activity and service wrapper processes. Ontologies are a suitable means to create a semantic description of an enterprise. Service ontologies, capturing the semantics of services in a more user-friendly and technology-independent way, are a promising technology to support service discovery and modeling. In contrast to service interfaces that build service abstractions which may hide specific properties of a service, an ontological service description has to cover all relevant effects of a service. Once potentially relevant services are discovered, they can be modeled as service activities. Therefore, we can rely on the service interface for service discovery at process runtime.

In addition to the service functionality, denoted by the service interface, the (expected) quality of the service to be used is an important characterization of a service and a service provider. Quality of service (QoS) includes, but is not limited to, classical non-functional parameters such as the throughput, response time, cost, reliability etc. Since in a VE environment several service providers may offer the same service, i.e., services implementing the same service interface, choosing the best implementation for the process that invokes the service is an important issue. It has to be emphasized that in general this decision cannot be made at specification time. Service providers may change the quality of their services over time and new service implementations may be offered. In addition, the notion of the “best” service may be relative to a specific MEP instance. For example, a MEP should use the fastest service implementation no matter what it costs, if the process is already close to the deadline. In contrast, if there is plenty of time left for the process to complete, a more cost-effective strategy can be chosen. To address these issues, each offered service has a set of QoS attributes attached that provide information on the expected QoS of the service execution. In addition, each activity variable in a process type that is bound to a service interface can have a description of the desired QoS. This description can include conditional statements to allow adaptation of the favored QoS to the current situation of a MEP instance. If there are multiple service providers, the provider should be selected whose service offer best matches the desired QoS goals.

Selecting a suitable service provider is called brokering and is performed by a dedicated system component, the service broker. The broker is a specialized server that in response to the request for providers of implementations of a service interface returns a list of available service providers plus information about the expected quality of service. The knowledge about service providers is gained by the broker through advertisements. The advertisements are constructed by service providers and sent to the broker.

4. Related work

Traditional WfMSs, e.g., FlowMark [21], InConcert [22], and COSA [27], can provide only limited support for applications involving partially defined processes, such as crisis mitigation. Dealing with dynamic aspects of processes is an emerging topic in the academic workflow research. An overview and a taxonomy for this problem area are provided in [19]. Existing work can be separated into approaches that allow for the dynamic modification of running processes and approaches that support a less rigid and/or descriptive workflow specification and therefore allow for more flexibility for the process participants. Dynamic situational roles to enable dynamic task force creation and awareness as provided by our approach are not addressed in any of these approaches.

ADEPTflex [31], WASA [35], Chautauqua [9], and WIDE [5] rely on a traditional predefined and static definition of workflow types and provide explicit operations to dynamically change running workflow instances. These change operations enable to add/delete activities and to change control and data flow within a running workflow instance while imposing constraints on the modifications to ensure structural integrity of the process instance. In contrast to our approach, process templates that cover a general crisis mitigation process and include explicit escalation points are not supported. To deal with dynamic situations, process participants are required to make ad hoc changes to a running process instance. This requires coordinators and domain experts in a crisis situation to be also experts in the respective process model, which is usually not the case. Furthermore, organizations that deal with large number of crisis mitigations tend to minimize or avoid ad hoc processes. Their experience and methods indicates that allowing ad hoc activities tend to foster a chaotic crisis mitigation process and this is not acceptable.

[18] and [16] provide for descriptive workflow specifications that are similar to our concept of process templates. These approaches enable the definition of flexible process types that cover a range of predefined process escalations. [16] introduces a high-level operator to capture cooperating activities that work on shared data. Thus, it is limited to a specific application domain. Mobile [18] allows for descriptive workflow specifications by providing control flow meta-modeling. The process designer can specify control flow patterns as needed by an application setting. Process participants, however, are treated as usual in traditional WfMSs. There is no dedicated support for coordinators and domain experts to escalate processes as runtime. Besides descriptive process specification, Mobile provides for late modeling of activities [18]. Our placeholder primitive includes a restricted form of late modeling to prevent chaotic process escalation.
[32] proposes a combination of descriptive workflow specification and dynamically changing running workflow instances. It focuses on architectural aspects of WfMS that address these issues, but it does not suggest any concrete process model primitives.

The term awareness has been used in many collaborative systems (not managed by a process specification) primarily to cover information about one's fellow collaborators and their actions [3, 30, 34]. However, only raw information is provided. This limited form of awareness is sometimes called telepresence [15]. One motivation for telepresence is that it allows users to readily determine who is available at remote locations so that ad hoc collaboration may be initiated [8]. Our notion of awareness largely subsumes the notion of awareness in collaborative systems both because: (1) we consider more than just user information, and (2) we leverage the process model to improve information relevance.

Commercial WfMSs and WfMC’s Reference Model [17] currently provide standard monitoring APIs. However, unless WfMSs users are willing to develop specialized awareness applications that analyze process monitoring logs, their awareness choices are limited to a few built-in options and process-relevant events. Besides monitoring capabilities, some WfMS offer primitives that enable process participants to subscribe to specific events that occur during process execution, e.g., execution of activities and activity deadline violations. InConcert [22], for example, provides e-mail notification of simple workflow conditions. FlowMark [21] supports similar notifications within its workflow client tool. Elvin is a general publish/subscribe framework [2] that could be considered event-based. However, no form of customized event processing other than filtering is performed. None of these systems provide mechanisms to cater the information for specific roles/classes of users, nor do they address the issue of combining information from multiple sources.

The preliminary consensus in research attempting to integrate process and service modes [1, 25, 28] is that traditional process technology as exemplified by current workflow standards [37, 33, 17] is not sufficient for integration of processes with heterogeneous services. However, related work provides only marginal extensions to capture services. Alonso et al. [1] suggest an architecture to implement MEPs that includes, for example, a component that documents semantics of external services. However, they do not provide a concrete process model to be used by the proposed architecture. Ludwig and Whittingham [28] and Klingemann et al. [25] suggest gateways to bridge enterprise boundaries. Gateways allow for the compensation of simple heterogeneities between service providers. In addition, Klingemann et al. [25] extends the activity model for services and provides so-called “control and notification events”. This idea is somewhat similar to CMM’s activity operations and application-specific activity states. However, they do not concretize their service model. In addition, it does not support application-specific activity states, subtyping of activity state types and service interfaces, and conversational coordination of services like CMM.

Application-specific activity states, which are a cornerstone in CMM for service modeling and conversational coordination, have been used also in several other workflow projects, but in ways that are virtually unrelated to our approach and not in a VE setting. We discuss them here for the sake of completeness. The MENTOR projects [38] relies on activity and state charts to model workflows. As a consequence, control flow within a process activity is specified by transitions between the states of the process. Therefore, process activity states are just a side effect of the use of the state chart formalism and are not visible outside the process. METEOR [26] permits application-specific states, but does not support state refinement and generalization as well as activity operations in order to deal with heterogeneous services what is vital in a MEP setting. Finally, the Mobile workflow model [23] provides application-specific activity operations together with a state transition diagram that defines valid activity operation invocation sequences. As in MENTOR, activity states are private to an activity. Thus, neither conversation coordination nor activity subtyping are possible.

Related work in service brokering deals with (service) contracts among a client and a service provider to capture the semantics of a particular service offer, cost-based service advertising and selection [14], as well as reliability issues of broker information [10].

Acknowledgments

Members of the CMI project at MCC, H. Schuster, D. Baker, A. Cichocki, T. Cassandra, and M. Rashid participated in the definition of the CMM model and its implementation described in this paper.

References


