Collaboration Policies: Access Control Management in Decentralized Heterogeneous Workflows

Mine Altunay
North Carolina State University/ECE, Raleigh, NC, USA
Email: maltuna@ncsu.edu

Douglas E. Brown, Gregory T. Byrd and Ralph A. Dean
North Carolina State University, Raleigh, NC, USA
Email: {debrown, gbyrd, ralph_dean}@ncsu.edu

***Abstract*—Service-oriented computing promotes collaboration by defining the standards layer that allows compatibility between disparate domains. Workflows, by taking advantage of the service-oriented framework, provide the necessary tools to harness services in order to tackle complicated problems. As a result, a service is no longer exposed to a small pre-determined homogeneous pool of users; instead it has a large, undefined, and heterogeneous pool of users. This paradigm shift in computing results in increased service exposure.**

The interactions among the services of a workflow must be carefully evaluated against the security risks associated with them. Classical security problems, such as delegation of rights, conflict of interest, and access control in general, become more complicated due to multiple autonomous security domains and the absence of pre-established trust relationships among the domains. Our work tackles these problems in two aspects: it provides a service owner with the necessary means to express and evaluate its trust requirements from a workflow (*collaboration policies*), and it incorporates these trust requirements into the workflow-planning framework (*workflow authorization framework*). Our policy-based framework allows bilateral peer-level trust evaluations that are based on each peer's collaboration policies, and incorporates the outcome of these evaluations into the workflow planning logic. As a result, our work provides the necessary tools for promoting multi-party ad-hoc collaborations, and aims to reduce the reluctance and hesitation towards these collaborations by attacking the security risks associated with them.

***Index Terms*—workflows, access control management, collaboration, heterogeneous, authorization**

I. Introduction

Grid computing [1] and service-oriented architectures [2] combine high-performance computing and storage services with high-bandwidth networks to provide a seamless computational environment that is not constrained by geographical or organizational proximity. One of the key challenges of such environments is complex applications that can (and often do) require services drawn from multiple administrative domains. These services do not share a pre-established framework for identification, authentication, and authorization. A loose collection of services may include services owned by rival companies, or separated by corporate firewalls, or otherwise inhibited from working collaboratively.

Workflows are often used to discover, harness, and manage this medley of services in order to tackle a complicated application. However, existing workflow frameworks are designed for a single homogeneous security model, where all services either belong to the same security domain or they already have pre-established relationships. As a result, the existing workflow security frameworks do not meet the needs of multi-domain workflows, namely heterogeneous workflows. Classical security problems, such as delegation of rights, conflict of interest, and access control in general, become more complicated due to these multiple autonomous security domains and the absence of pre-established trust relationships.

Furthermore, due to their multi-domain nature, heterogeneous workflows are often executed in a decentralized fashion [3, 4]. The inter-task dependencies are carried among the workflow participants without having a centralized workflow engine. Each participant service receives a self-describing workflow that expresses the tasks that must be performed by the receiver and the

Based on "Trust-based Secure Workflow Path Construction", by M. Altunay, D. Brown, G. Byrd and R. Dean, which appeared in the Proceedings of the International Conference on Service-Oriented Computing Amsterdam, NL, December 2005.

tasks that must be performed by the next participant service. Thus, data flows between the participant services directly, leading to peer-peer interactions. A centralized workflow, on the other hand, only allows data flow between the central workflow engine and each participant services. In addition to the heterogeneous nature of the collaborative computing environments, which lends itself naturally to decentralized workflows, the higher performance and scalability are among the other highly desired features of decentralized workflows.

Based on the characteristics of decentralized heterogeneous workflows, we have identified the following list of security requirements.

- *Peer-level mutual trust evaluations.* Complex workflows usually require several services to collaborate and interact with each other. These interactions often lead to unconsidered security consequences, such as direct or indirect accesses to a service by its peers. As a result, a service owner may have reservations about joining a proposed workflow due to the unexamined interactions. In order to address these reservations, each resource owner must be able to examine a proposed workflow graph and the interactions that it requires, and to evaluate its peers based on these interactions. A workflow framework that enables and incorporates the peer-level trust evaluations eliminates the unforeseen security violations, and reduces the reluctance towards workflow participation.
- *Decentralized workflow authorization framework.* Each service participating in a workflow may have domain-specific security policies and requirements that are confidential [5]. Thus, a workflow framework should have a decentralized access control model that leaves the final access decisions to each service. Moreover, the workflow model should not assume any prior knowledge about the security policies governing its services.
- *Context-based, collaboration-aware access control mechanisms.* Classical identity-based models or the families of role-based (RBAC) [6] and task-based (TBAC) [7] access control models assume that a service owner has prior knowledge of the user pool. This assumption is not adequate for today's highly dynamic, market-oriented web services paradigm, wherein the services are offered to anyone with the necessary credentials. Proposed access control models based on trust management [8] address this problem. However, trust management-based access control models still need to be incorporated with a high level abstraction that encompasses the requirements of multi-party collaborations. The new access control policies must be designed for evaluating access requests based on the context of a collaboration.

Our work proposes to address the above requirements by introducing collaboration policies and by incorporating the evaluation and enforcement of these policies into the workflow framework. The collaboration policy allows service owners to express access control requirements related to a multi-party collaboration. A collaboration policy expresses the required peer-level trust evaluations in order to offer a service to a workflow. Collaboration policies build on and extend existing service-level access control policies as an upper layer policy set.

Figure 1 shows a very high-level view of the stages involved in constructing and executing a workflow. In the Design and Choreography phase, a business process or application is expressed in terms of interacting tasks. In the Planning phase, the services which meet the operational requirements of the design are chosen, and the interactions among the services – in other words, the required collaborations – are identified. In the Execution phase, services perform their assigned tasks, and messages are passed among them to carry out the workflow operations.

In many workflow environments, security policies are

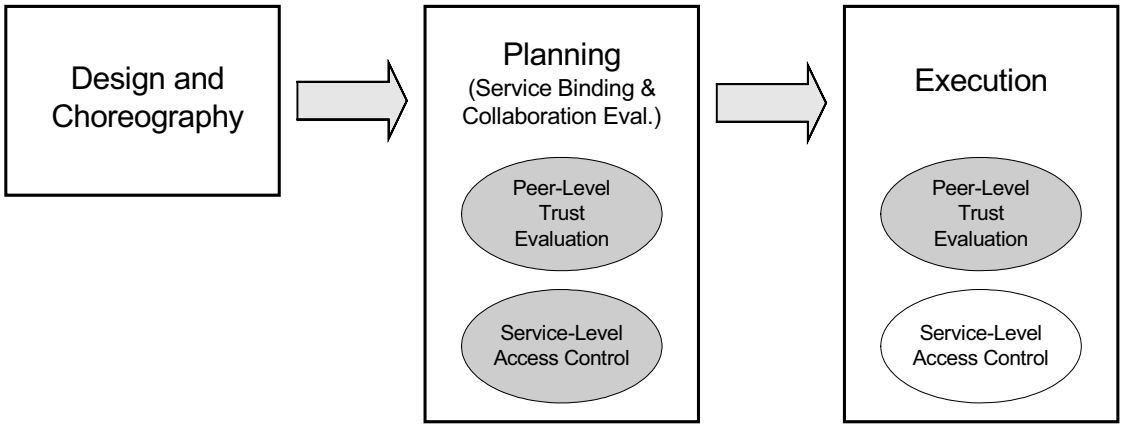

Figure 1. Phases of workflow construction and execution.

applied explicitly at the execution stage. When a service is invoked, it performs an access control check to authorize the invoker. There are multiple problems with this approach. First, the unidirectional nature of this access check does not sufficiently capture the service invoker's trust requirements. In other words, it does not allow a bilateral trust evaluation between a service invoker and the service. Second, in a multi-domain scenario, indirect service interactions may not be detectable and may cause security breaches, such as conflict of interest. Third, detecting security failures only at run-time leads to complete workflow failure, or to expensive re-planning and re-execution of workflow components.

Our framework incorporates the evaluation and enforcement of collaboration policies at the planning stage. It identifies the collaborations that compose a candidate workflow, and it enables the evaluation of peer-level interactions. Our framework provides services with the necessary means to express and evaluate their collaboration policies. At run-time, services may validate that the workflow interactions are consistent with the approved workflow plan. However, verification at the execution stage is beyond the scope of this work.

The primary components of our framework are a *workflow module* that can identify and consider the outcomes of peer-level trust relationships and a *policy module* at each service's domain that can evaluate and enforce the collaboration policies. The workflow module acts in a decentralized manner, in which each service is responsible for creating and enforcing its own access control policies. Our framework increases the availability of complex and trusted distributed environments for application communities that are unable or reluctant to create formal virtual organizations. It also provides the tools that support the complex trust relationships that already exist in these domains, allowing new opportunities for collaboration and scientific discovery.

II. Workflow Model

A workflow is modeled as a directed graph, where each vertex denotes a task and each edge denotes an inter-task dependency. Each task has a pool of available services that can functionally fulfill the requirements of that task. We assume that the discovery of services has been accomplished earlier during the discovery stage, and is beyond the scope of our work.

A service that is a candidate for performing a workflow task is called a Workflow Candidate Entity (WCE). A service is a higher-level abstraction that can provide computing power, storage, or simple remote code invocation. A service is the provision of any kind of facility to the public, and is not limited to its domain boundaries. We assume that the services are exposed over a network, and utilize the current Web Service standards such as WSDL, SOAP and UDDI. In addition, we model a service as a composite abstraction that may include executables, input and output files, and other objects that are accessed and used during service invocation. These objects that are included within a service abstraction can be viewed as a service's resources that are accessed for service utilization. This modeling of services as composition of multiple objects allows a policy administrator to create finer-grained access control policies for each service. For example, a subject that is not authorized for invoking a service's executable may be allowed for read access on output documents. The finer-grained policies are essential in multi-party collaborations where the subjects invoking a service and accessing the results of a service may be different. Our collaboration policy framework allows writing and enforcement of such finer-grained policies.

A workflow owner is an entity that requests a workflow in order to tackle a complex problem. The workflow owner contacts a workflow-planning engine, describes its problem requirements and collects the workflow results.

Our work focuses on selection of WCEs from each task pool with respect to their security and trust requirements. These requirements are not imposed by the workflow or any other central authority. Each WCE forms its own trust requirements (i.e. its collaboration policies) in order to join a workflow, and evaluates its peers to determine whether or not such requirements are met. Peer-level trust evaluations take place between the WCEs that belong to different task pool sets. After the completion of trust evaluations, the resulting workflow is a one-to-one mapping between tasks and services. We assume that each WCE within a task pool is equally desired from a functionality standpoint.

A WCE's collaboration policy is private, and is not divulged to other services, or to the workflow engine. Each WCE performs its own security evaluations and informs the workflow engine about its decision for participation.

Each WCE has access credentials that can be evaluated by its peers for authorization and authentication purposes. These credentials are assumed to belong to the actual service owner; thus, a WCE has the associated rights of its service owner. We model the WCE credentials similar to the proxy credentials [9] defined by Welch *et al.* such that a WCE has the same rights and privileges as that of the service owner.

Due to the heterogeneity of the services involved in a workflow, it is unreasonable to assume that an authorizing WCE has prior knowledge about the other peer services. As a result, password-based or role-based access control models are infeasible to utilize. We believe that trust-management models that tie a given credential set with an access decision are more suitable for dynamic ad-hoc workflows. Therefore, we assume that the attributes used in our collaboration policies must represent the characteristics of well-known credentials such as X.509 certificates.

Our framework expects each WCE to have an underlying access control policy and mechanism – co-existing with the collaboration policies. These underlying policies are traditionally designed to evaluate a standalone access request that is not part of a collaboration. The collaboration policies do not aim to

replace or modify the underlying access control policies or mechanisms. Rather, collaboration policies complement them as an upper layer policy set that is specially designed for access management in a multi-party collaboration. When desired, collaboration policies can re-use underlying access control rules by referring to the underlying policies.

A. Interaction Types

Based on a workflow graph, each service interacts with a number of peer services. These interactions can be examined in two categories: direct (bilateral) and indirect.

Direct interactions occur between WCEs that are distanced from each other by a single edge. Such WCEs are called direct neighbors. Any direct interaction between direct neighbors is a *bilateral relationship*, even when the flow of interaction seems to be one-sided. To illustrate this, refer to Fig. 2, where each workflow task is labeled by the services that perform the task. The simple collaboration scenario seemingly involves a one-sided relationship: Service A presents an input file to Service B for invocation and Service B determines if it trusts Service A for access. However, there are actually two relationships: (1) Service A determines that it trusts Service B and agrees to share a copy of its result file, and (2) Service B determines that it trusts Service A for invocation with the specified input file.

Both of Service A's and Service B's access requests involve risk. From Service A's perspective, Service B could be a rival company with whom A is not willing to share its results; from Service B's perspective, Service A could be a malicious user who sends a Trojan horse. Existing access control models such as TrustMaker [8], RBAC [6] or simple password-based schemes, are geared towards assessing the trustworthiness of the invoking party from the service's perspective. The *reverse* trust evaluation – i.e., the trustworthiness of the service from invoking party's viewpoint – is not explicitly modeled. Instead, it is assumed that the invoking party implicitly makes a trust evaluation before launching its request, most likely at service discovery stage. This implicit modeling may not work in a multi-party collaboration where services are not aware of each other until workflow execution time. Moreover, in a workflow context, a third party selects the participating services and requests them to interact. The selection of collaborating services does not necessarily equate to the existence of trust between the services. Therefore, each collaborating service must be allowed to perform bilateral authorization checks on each other.

Indirect interactions occur between WCEs that are distanced from each other by multiple edges (such as A and C in Fig. 2). These services are called indirect neighbors. The interaction between these parties occurs through intermediate services. Based on the sensitivity of a service, a service owner may put access control checks on indirect neighbors. There are several reasons why such indirect trust relationships must be carefully evaluated.

(1) Confidential documents or the results of a sensitive service are typically passed among several peers throughout a collaboration; thus even an indirect neighbor might have access to confidential data. Furthermore, partnership agreements and competition among businesses may prevent such interactions, even when they are safe from a security standpoint. In addition, peers that have security breaches in their domains may cause infection of their neighbors through these interactions.

(2) Indirect interactions can cause conflict of interest scenarios that cannot be caught by only inspecting the direct neighbors. An indirect neighbor involved with an access request may cause fraud and should not be allowed indirect access.

(3) A workflow chain may require or allow delegation of rights between WCEs. Delegated rights may have to travel through several disparate security domains and may be handled by intermediate peers until utilized for an access request. A WCE that receives an access request with delegated credentials may want to evaluate the trustworthiness of the intermediate parties as well as the peer that launched the access request. Likewise, the original owner of the delegated rights may also put trust requirements on the intermediate parties in order to prevent abuse of its rights.

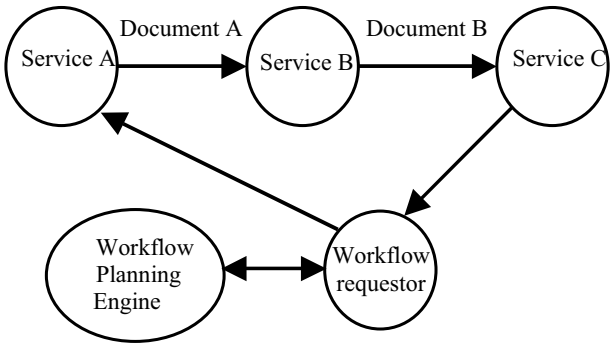

Figure 2. The Collaboration Scenario

B. Access Control Model

Our access control model uses the following terms as they are described. An object is an entity that belongs to a security domain. A subject is an entity that requests taking an action with an object. Access is a specified action that a subject may take with an object. An access right allows a subject to access an object. An authorized subject is a subject that has an access right for accessing an object. An access permission denotes an action that an authorized subject can take with an object. Each object has multiple access permissions.

An access request is a set of attributes that is sent to an object's security domain in order to access the object. It is represented as a set of {Subject, Object, Action}. Action indicates the permission that been requested over the object.

Each object has an access policy to evaluate an incoming access request. The policy may have multiple access rules, where each rule is a statement about the desired characteristics of an access request. An access

request that has the desired characteristics is authorized, i.e. it is accepted, and otherwise the request is denied.

III. The Overall Architecture

Our architecture combines enforcement of collaboration policies with workflow framework, as illustrated in Fig. 3. The workflow planning stage takes a workflow description from the preceding workflow stage (the description and choreography stage). For each task specified in the workflow description, the planning stage first discovers suitable services that are functionally capable of performing the task, then selects a single service for each task, and finally generates an executable workflow document with respect to the specified service bindings. The resulting workflow document is sent to the execution stage.

Our architecture incorporates the enforcement and evaluation of collaboration policies during the planning stage, indicated by the shaded boxes shown in Fig. 3. Two of our architecture components, Collaboration Locator Module (CLM) and Authorization Management Module (AMM), are combined into the workflow planning engine. The Policy Module (PM), on the other hand, is located at each peer service's domain. During the planning, the suitable services are discovered in the same manner as explained above. However, rather than generating service-task mappings immediately, the set of discovered services are sent to CLM for trust evaluations. CLM receives a set of discovered services for each workflow task. It selects one service from each set to start the peer-level trust evaluations. CLM contacts each selected service's PM, and notifies them about the proposed workflow plan. Each PM requests information from CLM in order to carry out its peer-level trust evaluations, such as the identity and location of its peers. CLM generates the necessary information from the workflow graph for each PM, and passes it onto AMM.

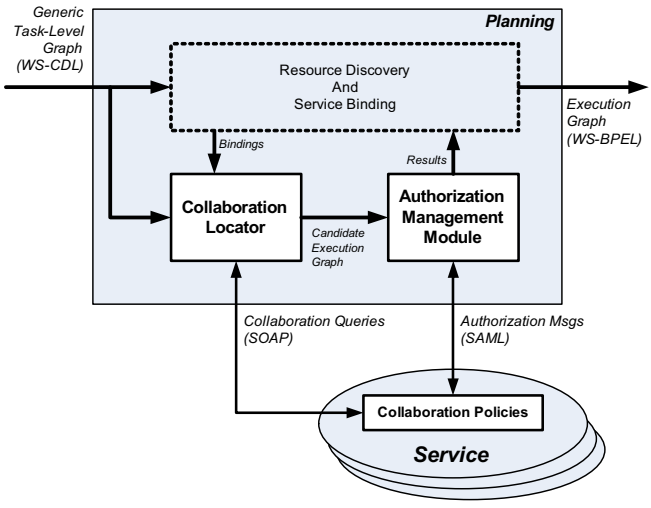

Figure 3. The Proposed Workflow Planning Architecture.

AMM notifies each PM with the generated information, and waits for the result of each PM's decision on joining the workflow. Each PM carries out the trust evaluations with its peers based on the collaboration policies, and reaches a policy decision that states whether or not the proposed workflow constitutes a security threat. The decision of each PM is sent back to AMM so that AMM can determine the feasibility of proposed task-service mappings. For each task that is been declined by a candidate service, AMM contacts CLM and requests a new service that is functionally capable of performing that task. CLM selects a new service, generates the necessary information for this new service, and passes it onto AMM. AMM contacts the new service's PM about the proposed workflow and waits for a policy decision. The new service carries out its trust evaluations and returns a policy decision. In the meantime, AMM also contacts the services that are affected by the selection of a new service, and notifies them about the replacement. These services evaluate the new service and may change their policy decision accordingly. Until a desirable task-service mapping for each task is found, approved by both the service owner and AMM, the same steps are repeated by AMM. Once the entire service-task mappings are approved, an executable workflow document with the corresponding service bindings is generated and sent for execution.

IV. Collaboration Policies

A. Policy Language

A collaboration policy is an upper layer access rule collection, co-existing with already defined underlying access control policies. A collaboration policy states the trust requirements sought from each collaborative peer in order to provide a service to a workflow, and is used to evaluate access requests originated from within a collaboration. For each service that is offered to a workflow, there must be an individual collaboration policy that states the requirements for granting access to the service.

Definition 1: Collaboration Policy CP is a collection of access control rules, each of which represents the trust requirements sought from a collaborative peer based on the peer's interaction with a requested service. The Boolean logic operators may be used to combine the results of access rules. A policy decision is either "access" or "deny".

A key point of the collaboration policy model is that a service owner is allowed to exercise control over access from indirect and direct neighbors. The collaboration policy allows specifying access control rules that vary with respect to the interaction types. In other words, it is possible to define multiple access rules, each of which is designed for a different interaction type, for granting access to a single service. As a result, multiple peers, each with a different interaction type, may be evaluated against a different access rule, and the logically combined results of these rules determine whether the access must be granted to the proposed workflow or not.

Definition 2: Access Rule AR = {Target, Type, Conditions}, where Target indicates the peer, object, and action to which this rule applies, Type indicates the type

of the access rule, and Conditions contains a collection of access conditions. The access rule evaluates to either "access" or "deny".

The Target element is also a set with three members: *PeerLocation, Object,* and *Action*. The *PeerLocation* element indicates the type of interaction a peer service should possess in order to be evaluated against the access rule. The *PeerLocation* element can be represented in two ways: either as a *direction:interaction* pair, or by a keyword *EndUser*. In the former case, the direction could either be upstream (up) or downstream (down). The interaction element indicates the interaction type that must exist between the authorizing WCE and the requesting peer service. The interaction element is either one of the keywords *direct*, *indirect*, and *any*, or it could be an integer. The keywords *direct/indirect* respectively state that only a requesting peer with a direct/indirect interaction can be evaluated against the access rule. The keyword *any* states that any peer, regardless of its interaction type, must be applied to this rule. When the interaction is an integer, it indicates the distance (number of edges) between the authorizing WCE and the requesting peer service. Instead of a *direction:interaction* pair, the keyword *EndUser* may be used. *EndUser* indicates that the workflow owner must be evaluated against this rule regardless of its interaction type.

Definition 3: Target = {PeerLocation, Object, Action}. Target element determines whether an access rule AR must be applied to an incoming access request. PeerLocation indicates the interaction type or distance of the requesting peer, Object indicates the requested entity, and Action indicates the requested permission over the Object.

Definition 4: PeerLocation, indicates the interaction type and the relative location of a requesting peer with respect to the authorizing WCE. It is represented either as a direction:interaction pair or by the keyword EndUser

Definition 5: Direction indicates the relative location of a requesting peer with respect to the authorizing WCE. It is either upstream (up), or downstream (down).

Definition 6: A requesting peer is in the Upstream Direction of a WCE (indicated by vertex V) when there exists a directed walk $W= v_0, e_1, v_1, ..., e_n, V$ *such that the requesting peer is a member of W, V represents the authorizing WCE, and the* e_n *is a backward edge of* V.

Definition 7: A requesting peer is in the Downstream Direction of a WCE (indicated by vertex V) when there exists a directed walk $W = V, e_1, v_1, ..., e_n, v_n$, *such that the requesting peer is a member of W, V represents the authorizing WCE, and the* e_1 *is a forward edge of* V.

Definition 8: The interaction indicates the interaction type of a requesting peer. It is one of the keywords direct, indirect, and any, or it can also be specified as an integer. When specified as an integer, it indicates the distance between a requesting peer and an authorizing WCE.

If an access rule does not have a *Target*, the *Target* specified within its *Conditions* element is applied. (The *Conditions* may iteratively include other access control rules; each may have a separate *Target* element.) If the *Conditions* element has no *Target* specified, then the access rule becomes inapplicable to any access request. If a rule has a *Target* set, but some elements of the *Target* are not specified (null), then the access requests that are matching the remaining elements of the *Target* set are applied to this access rule. When an access rule has a specified *Target*, and it also contains additional *Targets* specified within its *Conditions* element (i.e. the *Conditions* iteratively includes other access rules), each access request is first checked against the encapsulating access rule and then against the other *Targets* specified within the *Conditions*.

The *Type* of a rule can be one of the keywords *L, U,* D_D, D_U. A detailed discussion of access rule types is given in section IV.B.

The *Conditions* element of an access rule is composed of multiple access conditions that may be logically combined by Boolean logic operators. In addition, *Conditions* element may iteratively contain access control rules that are used to express complicated access conditions. The access control rules may also be logically combined with Boolean logic operators. The result of the *Conditions* element determines the outcome of an access rule.

Definition 9: An Access Condition (AC) states the desired attributes of an access request for granting access, and tests an incoming request against these attributes. A collection of logically combined predicates may be used for expressing a single access condition. An access condition may result in "true" or "false". A true evaluation is associated with "access", and false evaluation is associated with "deny"

Definition 10: A Predicate, P(e), is a Boolean-valued function, where e denotes a variable. The variable represents an entity associated with an access request such as a subject, an object, or an action. A predicate function states the desired attributes of a variable, and evaluates to true when such attributes exist.

Definition 11: An Attribute is a characteristic of an entity, such as a subject, an object, or an action. Each attribute has a name and a value.

A predicate function uses several attributes, each of which belongs to a separate entity that is associated with an incoming access request. Sometimes, the entities may all have the same attribute with different values. In order to distinguish between the entities and their corresponding attributes, the predicate function uses one of the keywords Subject, Object, or Action to state that which entity's attribute is required.

Example 1:

```
AC: {
      (Subject.X509OrgNameAttribute =
      "OrganizationNameX")
    }
```

Above access condition *AC* has a single predicate stating that an access request must have a subject entity such that it must have an attribute named

X509OrgNameAttribute and its value must equal to "OrganizationNameX". For brevity, we just present a simple predicate function with an equality symbol in this example. However, it is possible to define more complex predicate functions that take multiple attributes as input.

The attributes used within a collaboration policy must easily be demonstrated by a requesting peer, and be easily evaluated by the policy engine. Widely-used credential types, such as X.509, may be used, in order to provide uniformity and compatibility, as illustrated in Example 1.

B. Access Rule Types

There are four types of access rules: *L, U, D_D, and D_U*. *U* stands for underlying, and allows re-use of an underlying access control rule. *L*, on the other hand, stands for local and indicates that the access rule is evaluated only in the context of the current collaboration policy. D_D and D_U indicate that the access rule deals with delegated credentials.

Definition 12: A type L access rule indicates that the access conditions of the rule is contained in the local collaboration policy.

Definition 13: A type U access rule has an empty Conditions element. An access request that matches the Target of this access rule must be evaluated against an underlying access rule (UAR) that matches the access request.

Recall that in addition to the collaboration policies, each WCE is expected to have an existing underlying access control policy (*UP).* When desired, a policy administrator can re-use the underlying access control rules (*UAR)* defined within the *UP*. Utilization of *UAR* indicates that the access request must be evaluated as if it is not part of a collaboration. The result of *UAR* determines the result of the *AR*.

During the enforcement of a type *U* access rule, the collaboration policy module contacts the underlying policy engine of *UP*. The collaboration policy module creates a new access request based on the original access request that it received. While creating the new access request, the policy module may eliminate some of the information that is not meaningful to the underlying policy engine, such as the interaction type or the direction of a requesting peer. This can be done by replacing the unnecessary information by the information that is required by the underlying policy model, which can also be extracted from the original access request.

Example 2:

```
CP1 : {
        AR { {up:direct, Service C, execute},
                U, Conditions: ∅}
        AND
        AR { {up:indirect, Service C, execute},
                U, Conditions: ∅}
}
```

CP_1 states that any direct or indirect upstream neighbors of Service C must be evaluated against the corresponding *UAR*.

Example 3:

```
CP2 : {
        AR { { up: direct, Service C, execute }, U,
                Conditions: ∅}
        AND
        AR { { up: indirect, Service C, execute},
                L, Conditions:
                    {
                        (Subject.X509OrgAttr =
                        "Organization Y")
                    }
        }
}
```

CP_2 states that all upstream direct neighbors of Service C must be evaluated against the corresponding *UAR*, whereas, the upstream indirect neighbors must be members of "Organization Y".

Definition 14: Rule type D_U indicates that accessing the requested Object with delegated rights is allowed as long as the Conditions element of this rule evaluates to true. The access rule may include an optional element, re-delegation distance (Re-delDist). When specified, the Re-delDist is located between the Type element and Conditions element, causing the access rule to have four elements instead of three.

Definition 15: Re-delegation distance Re-delDist denotes the number of times the rights may have been re-delegated among the intermediate peers.

Type D_U access rules are designed to evaluate the access requests that are launched with delegated rights. An access request with delegated credentials is denied access if there exists no matching rule of type D_U, indicating that the object owner does not allow access with delegated credentials. A type D_U rule may define varying access conditions on the intermediate parties that are involved with the delegation. Re-delDist indicates the acceptable number of times the rights may have been re-delegated between the intermediate peers. An access request with delegated rights that have been re-delegated more times than what Re-delDist denotes is denied access. When Re-delDist is not specified, the acceptable re-delegation distance would be infinity.

Definition 16: Rule type D_D indicates that the downstream delegation of rights is allowed as long as the Conditions element of this rule evaluates to true. The Object element of Target set denotes the rights that are being delegated and the Action element of the Target set is always set to "delegate". The access rule may optionally include an element indicating the re-delegation distance (Re-delDist).

Type D_D access rules are designed to regulate the delegation of rights, and thus to prevent the abuse of rights by unauthorized parties. The access conditions define the authorized peers that may be allowed to obtain

the delegated rights. Based on the re-delegation distance, access conditions may be defined over multiple peers that are involved with the delegation. The access request for a type D_D rule is made by the original rights owner, and the request states the rights owner's desire to delegate its rights. The actual delegation of rights is dependent on the access rule outcome. The absence of a type D_D rule means that the delegation of the rights is not allowed.

Example 4:

```
CP3: {
    AR { {up:any, Service C, execute},
        DU, 0, Conditions:
            {
                AR { {up:direct, Service C,
                execute}, L, Conditions:
                    {
                    (Subject.X509OrgAttr
                    = "Organization Y")
                    }
                }
                AND
                AR { {up:indirect, Service C,
                execute}, U, Conditions: ∅}
            }
    }
    OR
    AR { {up:direct, Service C, execute},
        U, Conditions: ∅}
}
```

CP_3 has two access rules: a type U rule and a type D_U rule. CP_3 allows access in two cases: either the direct neighbor must be authorized for Service C (indicated by the type U rule), or an indirect authorized neighbor must delegate its rights to the direct neighbor (indicated by the type D_U rule). In addition, the Re-delDist indicates the acceptable re-delegation distance as zero, i.e. re-delegation is not allowed. During the peer-level trust evaluations, this policy checks the credentials of its direct and indirect neighbor against these two rule types. If the type D_U rule is satisfied, (whereas, type U rule is not satisfied) the policy returns a conditional "access", stating that at run time, the indirect neighbor must also agree to delegate its rights to the direct neighbor. Note that at evaluation of CP_3, it is unknown whether the authorized indirect neighbor is willing for such a delegation or not; CP_3 only discovered that existence of the authorized upstream neighbor.

Example 5:

```
CP4: {
    AR { {down:any, attrCert, delegate}, DD, 0,
        Conditions:
            {
                AR { {down:direct, attrCert,
                read }, U, Conditions: ∅ }
                OR
                AR { {down:direct, attrCert,
                read }, L, Conditions:
                    {
                    (Subject.X509Name
                    = "Alice")
                    }
                }
            }
    }
}
```

CP_4 states that the downstream delegation of *attrCert* is allowed as long as the downstream direct neighbor is "Alice" or the direct neighbor is authorized by the corresponding *UAR*. The re-delegation is not allowed.

C. Policy Module

Each WCE must have a Policy Module (PM) that can enforce the collaboration policies, and communicate with other peers and the workflow engine. The policy module works in two modes: enforcement and request. The enforcement mode evaluates incoming access requests against the peer's collaboration policy, and determines the policy outcome. The request mode, on the other hand, creates and sends access requests to the other peers so that mutual trust evaluations can be carried out.

In the enforcement mode, the policy engine retrieves the collaboration policy of a service. The PM determines the interaction types that are required by the service's policy. Before starting the trust evaluations, the policy engine must state the required interaction types to the workflow engine. The planning engine (CLM, discussed in Section V.A) identifies all the peers in the workflow graph that possess the requested interaction types with the offered service. This set of identified peers constitutes the upstream and downstream access chains of a service. A service's policy module must receive individual access requests from each of these peers within its access chain. Each access request may correspond to multiple access rules within a collaboration policy. The policy module must evaluate each incoming access request against all matching access rules. The policy module logically combines the outcome of each access rule and determines the policy outcome. A missing access request is reported to the workflow engine (AMM, discussed in Section V.B). If the access request does not arrive within an allotted time, the access rules involved with the request is evaluated to "deny".

When the policy module determines the outcome of its policy, it reports this result back to the workflow engine (AMM). A policy may have two outcomes: "access" or "deny". "Access" is interpreted as a participation in the workflow, whereas, "deny" is interpreted as the rejection of the proposed workflow. The policy outcome may be

dependent on a condition. In this case, the policy module determines the conditions that must be satisfied in order to join the workflow, and sends them to the proposing workflow engine. These conditions are checked by the workflow engine (AMM) for consistency with other participants' conditions and achievability. The policy module, however, does not completely rely on the workflow engine for enforcing its conditions. During workflow execution stage, the policy module may re-enforce the collaboration policies, and deny access requests that do not meet the stated conditions. The run time re-enforcement is beyond scope of this work.

In the request mode, the policy module sends access requests to other peers, and allows them to evaluate its access rights associated with the other peers' services. The workflow engine informs the policy module about the peers that require the trust evaluations. The policy module must obtain the locations of authorizing peers' policy modules in order to send access requests. (Note that there might be multiple peers that require bilateral evaluations.) In addition, the policy module must be informed about the interaction types that it possesses with respect to each authorizing peer, the requested objects and the actions on these objects. The workflow engine determines all of this information from the workflow graph and sends it to the policy module. The policy module generates an individual access request for each authorizing peer, and sends it to the corresponding policy module.

V. Workflow Authorization Framework

We incorporate the peer-level trust evaluations into the planning logic; therefore we eliminate possible run-time security failures that are caused by failing to meet collaboration policies. Our workflow module has two components: Collaboration Locator Module (CLM) and Authorization Management Module (AMM).

A. Collaboration Locator Module (CLM)

CLM takes a workflow graph described in an XML-based choreography language, WS-CDL [10], and accompanying task pools (i.e. sets of discovered services) as its inputs. CLM triggers the peer-level trust evaluations. It picks a WCE from each task pool randomly, since all WCEs are assumed to be equally desired from a functionality standpoint. CMM contacts each WCE's policy module, and obtains the interaction types required by its collaboration policies. CLM generates the access chains for each WCE accordingly. The upstream access chain of a WCE is a subset of the in-tree of vertex v such that WCE performs the task denoted by v. Likewise, the downstream access chain of a WCE is a subset of the out-tree of vertex v such that WCE performs the task denoted by v. (In-tree of a vertex represents the paths of a graph that lead to a given vertex. Out-tree of a vertex represents the paths of a graph that originates from a given vertex.) Note that the actual in-tree and out-tree of a WCE may include peers that do not posses the requested interaction types; therefore, the access chains must be shortened to exclude undesired peers. CLM sends the access chains to AMM.

B. Authorization Management Module (AMM)

There are two key responsibilities of AMM: (1) collecting WCEs' decision on the proposed workflow and checking the consistency between these decisions, and (2) replacing a WCE which has declined a workflow or has a condition on its participation that cannot be satisfied by the current workflow.

AMM is also responsible for sending the necessary information to each participant so that it can start its peer-level trust evaluations. AMM sends two sets of information to each WCE's policy module; the first set includes the access chain of the WCE, the second set includes information about the peers that require evaluating the receiving WCE. In other words, the first set includes the identities and the interaction types of the peers that must be authorized by the receiving policy module. The second set includes the identities and the interaction types of the peers that the receiving module must be authorized by. Upon receiving these two sets of information, each participant's policy module either evaluates its peers, or creates access requests in order to be evaluated by its peers. At the end of the mutual trust evaluations, each policy module reaches a decision on joining the workflow and states this decision to AMM.

AMM collects the outcomes of collaboration policies from each WCE. A WCE that declines to join the workflow is replaced with another WCE from the task pool. AMM states the decline decision to CMM along with the task name that needs a new WCE. CMM selects a new WCE from the corresponding task pool, obtains the new WCE's requested interaction types, generates a new access chain, and pass it onto AMM. AMM follows the same steps and prepares two sets of information for this new WCE in the same way before. In order to update the other peers about this replacement, AMM sends the identity and the interaction type of this participant to the peers that either had already requested evaluating the new peer based on its interaction type, or this new WCE requests to evaluate. Each WCE that receives the update information either creates a new access request and sends it to the new WCE, or evaluates the new WCE's access request against its own collaboration policy.

If a WCE agrees to join a workflow on a conditional basis, the AMM checks the conditions and determines their feasibility. Once a suitable WCE for each workflow task is found, the AMM binds these participants to the specific workflow tasks, and outputs it to the next workflow stage.

Recall that each WCE may state a conditional decision on the proposed workflow participation. There could be multiple WCEs with conflicting conditions. Therefore, AMM must examine each decision and determine whether the conditions can be mutually satisfied. A simple message format must be designed for expressing the conditions. Our message format is shown below.

Definition 17: Participation Decision = {Decision, DecisionConditions: { ReqUpDelDist, DownDelDist} }, where decision could be either "access" or "deny", and DecisionConditions is a set of two integers. An unspecified condition set means that the WCE neither

grants access with delegated credentials, nor delegates its own credentials.

ReqUpDelDist stands for required upstream delegation distance. It is an integer that is used when access with delegated credentials is allowed. This integer indicates that an authorized upstream peer has been found at a location that is distanced by ReqUpDelDist edges from the authorizing WCE. A condition set with a non-zero ReqUpDelDist states that the peer indicated by this integer must allow downstream delegation of its rights. Otherwise, the access would not be allowed.

Likewise, DownDelDist stands for downstream delegation distance. It is an integer that is used when downstream delegation of access rights are allowed. The DownDelDist is used when an authorizing WCE discovers a downstream peer that is trustworthy for delegation. It is typically used between peers that have business partnerships to support each other.

Example 7:
Assume A, B, and C correspond to three distinct WCEs, where A-B and B-C are direct neighbors and A-C are indirect neighbors with a distance of two. Assume also the following participation decisions are received from each WCE:

Participation Decision C: {access, DecisionConditions: {2, 0}}. This message shows that A is an authorized peer for C's service and if A allows downstream delegation of its rights, C would accept an access request from B with the delegated credentials. C does not allow downstream delegation of its own credentials.

Participation Decision A: {access, DecisionConditions: ∅}. A is willing to join the workflow. However, A neither accepts delegated credentials or allows delegation of its credentials.

Participation Decision B: {access, DecisionConditions: ∅} which shows that B participates the workflow with no conditions.

Having looked at all the messages, AMM determines that the conditions are inconsistent and the proposed graph is infeasible for execution. AMM should also determine which peer to replace so that feasible task-service assignments can be found.

The second key responsibility of AMM is to replace a WCE with infeasible participation conditions from the workflow graph. A new WCE must be retrieved for the workflow task. There are several important issues to consider while replacing an unfit WCE. (1) The number of neighbors of the unfit WCE. The high number of neighbors indicates that replacement is going to cause many trust re-evaluations. Therefore, it is better to select a WCE that has few number of neighbors. (2) The collaboration policy of an unfit WCE. Selecting an unfit WCE that has a more restrictive collaboration policy (i.e. high number of conditions, long access chain) than other unfit WCEs would decrease the number of trust re-evaluations. The more restrictive collaboration policy may also cause multiple conflicts with other WCEs. We believe that design and implementation of a efficient replacement algorithm is crucial for the success of workflow-planning stage, and it is among our future works.

VI. Related Work

There are several workflow authorization frameworks proposed [11, 12, 13]. These frameworks are designed to manage workflow authorization within a single authorization domain (such as a large organization), and focus on defining authorization constraints over workflow tasks in order to regulate which service or human subject can perform a task. Since these frameworks have a single-domain model, they omit peer-to-peer trust relationships. Every workflow participant is modeled to be a member of the same security domain, and there is established trust between the participants. Furthermore, the policies governing the security domain are trusted by all the participants, and consequently, none of them has separate policies to protect themselves. One of the contributions of the above frameworks is to synchronize accesses over workflows tasks with respect to the workflow progression. Therefore, no workflow participant can execute a task before the workflow reaches a certain state.

Other approaches [14, 15, 16] are similar to the above frameworks in that they use authorization constraints over workflow tasks; however, they rather focus on extending RBAC models to express the authorization constraints.

Kang et al. [17] recognizes the inter-organizational, nature of new-generation workflows, and adopts a multi-domain security model. However, Kang's model requires a pre-established trust relationship between the disparate security domains. Kang uses the RBAC model, and assumes that a central workflow engine can access each peer's security policy in order to generate role-mappings across the domains. Therefore, the workflow engine can determine which peers may interact with other peers. Kang does not discuss the details of the role-mapping process, which requires existing role ontology between the domains. The main drawback of Kang's work is that it does not allow building dynamic ad-hoc workflows, where the workflow engine should not require prior knowledge about the internal security policies of participants.

Koshutanski et al. [5] proposes an authorization framework for ad-hoc workflows. Based on a workflow owner's request, a workflow engine dynamically selects suitable services to perform each task. It assumes that none of the services publicly announces their access control policies. Koshutanski's key contribution is an authorization mechanism between the workflow owner and the services. Instead of sending access policies at the data-level (i.e., publicly exposing the policies), each service sends a mobile process to the workflow owner. The mobile process must be executed in the workflow owner's domain, and it determines whether the workflow owner is authorized for the requested service. The mobile code is assumed to access the credentials stored within the workflow owner's domain so that an access decision

can be reached. The reliance on mobile code introduces other security issues, such as how the workflow owner's domain can verify the mobile code, and how the mobile code should retrieve all the required credentials. Koshutanski's framework neglects the peer-level trust evaluations, and focuses on authorizing the workflow owner to each service individually.

The WAS framework [18] adopts a multi-domain security model and targets grid-based computing environments. WAS, like Koshutanski's work, focuses on determining the access policies associated with a service that is suitable to perform a workflow task. However, unlike Koshutanski, it assumes pre-established trust relationships between the domains. The WAS engine functions as a trusted third party between the service owners and the workflow owner. Each service owner informs the WAS engine about its services, and delegates the associated access rights to the WAS engine. Upon building a workflow, the WAS server determines which services, and accordingly which credentials, are necessary for execution. The WAS engine delegates these rights to the members of the grid community. WAS framework may function well in small grid communities, where prior trust and community-wide policies can be established. However, it is not well suited for ad-hoc heterogeneous workflow models.

Recent work by Shehab [19] also addresses security issues of multi-domain collaborations. This framework assumes that there are cross-domain role mappings and each domain is aware which mappings are forbidden or authorized. In addition, formation of the cross-domain mappings is not within the scope of their work, and assumed to be handled priorly. Their approach is focused on tracking the history of an access request, which is a list of the domains that are visited until the access request. All the domains that are involved with an access request can be evaluated at access decision time. Their approach is similar to ours in that they allow a workflow participant to check the direct or indirect domains involved with an incoming access request. However, they differ in their assumption of cross-domain role-mappings. They state that domains must somehow consult each other and generate role-mappings. Our proposed framework does not require pre-defined mappings between domains, formation of which require exposure of policies between the domains and an established trust relationship. Instead, our approach serves to form these cross-domain role mappings without the public disclosure and in the absence of established trust relationships. As a result, their framework has a rather straightforward approach at detecting illegitimate access requests that traveled through unauthorized domains; each service has a complete list of forbidden mappings between domains and an access request that includes a forbidden link is denied access.

VII. Conclusion

Dynamic heterogeneous workflows are ideal collaboration environments, in which otherwise unsolvable tasks can be tackled by combining available services from different domains. The users can harness services on-demand, based on their needs, and avoid issues, such as software writing and maintenance, low performance, and storage, which can all be handled by individual services. However, the highly appreciated benefits of collaboration come with the cost of more complicated security and trust problems. From the workflow-planning standpoint, dynamically selecting services, incorporating their willingness for participation, and keeping the consistency among each peer's participation decision are not trivial. From the service owner's standpoint, on the other hand, collaborations present unique challenges; determining appropriate trust evaluations and enforcing policies to prevent security breaches that are unique for multi-party collaborations are just a few of them. Nevertheless, without the underlying technologies that address these security and trust issues, collaboration will never reach its true merits.

We have presented a policy language and its enforcement that allows bilateral peer-level trust evaluations. We have also shown how the result of these evaluations may be incorporated into the workflow-planning framework so that otherwise undetected security failures may be eliminated; thus the reluctance towards workflows may be reduced.

We believe that the paradigm shift happening in other aspects of computing society is also having its impact on the well-established models used for security and access control management in particular. The progression of ubiquitous, on-demand nature of services will manifest collaborations as a requirement, rather than a desired feature. We believe that our framework is one of the first steps towards a more sophisticated generation of access control models that can finally embrace the paradigm shifts happening in other aspects of computing.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations." *International J. Supercomputer Applications*, 15(3), 2001.

[2] World Wide Web Consortium (W3C), "Web Services Architecture." http://www.w3.org/TR/2004/NOTE-ws-arch-20040211, 2004.

[3] G.B. Chafle, S. Chandra, V. Mann, and M.G. Nanda, "Decentralized Orchestration of Composite Web Services" in *Proc. of 13th Intl. ACM Conference on World Wide Web, 2004*, pp. 134-143.

[4] V. Atluri, S. A. Chun, and P. Mazzoleni, "A Chinese Wall Security Model for Decentralized Workflow Systems", *In Proc. of 8th ACM Conf. on Computer and Communications Security*, 2001, pp. 48-57.

[5] H. Koshutanski and F. Massacci, "An Access Control Framework for Business Processes for Web Services." ACM Workshop on XML Security, 2003, pp. 15-24.

[6] R. Sandhu, "Role-Based Access Control Models." IEEE Computer (1996) 29(2):34-47

[7] R.K. Thomas, R. Sandhu, "Towards a Task-based Paradigm for Flexible and Adaptable Access Control in Distributed Applications." ACM SIGSAC New Security Paradigms Workshop (1992-93) 138-142.

[8] M. Blaze, J. Feigenbaum, J. Ioannadis, A.D. Keromytis, "The Role of Trust Management in Distributed Systems Security." In Secure Internet Programming: the Security

Issues for Mobile and Distributed Objects. Springer-Verlag, 1999, pp. 185-210
[9] V. Welch, et al., "Security for Grid Services." 12th Intl. Symp. on High Performance Distributed Computing, 2003.
[10] World Wide Web Consortium, *Web Services Choreography Description Language Version 1.0*, 9 Nov 2005, http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/
[11] V. Atluri and W-K. Huang, "An Authorization Model for Workflows." Fifth European Symposium on Research in Computer Security, 1996, pp. 44-64.
[12] W-K Huang and V. Atluri, "SecureFlow: A Secure Web-enabled Workflow Management System." 4th ACM Workshop on Role-based Access Control, 1999.
[13] K. Knorr, "Dynamic access control through Petri net workflows." 16th Annual Conference on Computer Security Applications (ACSAC'00), 2000, pp. 159-167.
[14] E. Bertino, E. Ferrari, and V. Atluri, "A Flexible Model Supporting the Specification and Enforcement of Role-Based Authorizations in Workflow Management Systems." 2nd ACM Workshop on Role-Based Access Control, 1997.
[15] K. Tan, J. Crampton, and C.A. Gunter, "The Consistency of Task-Based Authorization Constraints in Workflow Systems." 17th IEEE Computer Security Foundations Workshop (CSFW'04), 2004, pp. 155-169.
[16] P.C.K. Hung and K. Karlapalem, "A Secure Workflow Model." Australasian Information Security Workshop Conference, 2003, pp. 33-41.
[17] M.H. Kang, J. S. Park, and J. N. Froscher, "Access-Control Mechanisms for Inter-Organizational Workflow." Sixth ACM Symposium on Access Control Models and Technologies, 2001, pp. 66-74.
[18] S-H. Kim, J. Kim, S-J. Hong, and S. Kim, "Workflow-based Authorization Service in Grid." Fourth International Workshop on Grid Computing (GRID'03), 2003, pp. 94-100.
[19] M. Shehab, E. Bertino, and A. Ghafoor, "Secure Collaboration in Mediator-Free Environments." ACM Conf. on Comp. and Communications Security, Nov. 2005, pp. 58-67.

Mine Altunay is currently pursuing her PhD in computer engineering at North Carolina State University (Raleigh, NC, USA). She received a BS in electrical and electronics engineering from Bilkent University (Ankara, Turkey) in 2001. Her studies focus on trust and security issues in distributed heterogeneous environments and service-based workflow management systems.

Douglas E. Brown received a multidisciplinary BS from Framingham State College (Framingham, MA, USA) focusing on applications of artificial intelligence to educational systems. He is the bioinformatics lead at North Carolina State University's Center for Integrated Fungal Research (Raleigh, NC, USA). Prior to joining NCSU, he was an industrial consultant in a range of fields including flexible factory automation, secure Unix kernels, human computer interfaces, and participated in creating international standards. His research interests include informatics analysis and modeling of cellular systems, information visualization, and high throughput computing.

Gregory T. Byrd received a BS in computer engineering from Clemson University (Clemson, SC, USA) in 1984. He received MS and PhD degrees in electrical engineering from Stanford University (Stanford, CA, USA) in 1985 and 1998, respectively.

He is an Associate Professor of Electrical and Computer Engineering at North Carolina State University (Raleigh, NC, USA). Prior to joining NC State, he was Principal Scientist at Celotek Corporation, and a Member of the Technical Staff at MCNC and the NC Supercomputing Center. His research interests include network security, grid computing, parallel computer architecture, and network processors.

Dr. Byrd is a Senior Member of the IEEE.

Ralph A. Dean holds an undergraduate degree in botany from the University of London's Imperial College and a Ph.D. in plant pathology from the University of Kentucky. He is a William Neal Reynolds Distinguished Professor of Plant Pathology and Director of North Carolina State University's Center for Integrated Fungal Research. His research focuses on discovering and analyzing the function of genes from economically important fungi and the crops they affect. Dr. Dean holds several patents and has received numerous awards for his research. In 2004, Dean was awarded the Huxley Memorial Medal from Imperial College and the U.S. Department of Agriculture Honor Award. His work has been published in top international journals, including a recent article in the journal Nature resulting from his leadership of an international effort to sequence the genome of the rice blast fungus.