Accepted on July 14, 2008

International Journal of Software Engineering and Knowledge Engineering
© World Scientific Publishing Company

World Scientific
www.worldscientific.com

ONTOLOGY-GUIDED SERVICE-ORIENTED ARCHITECTURE COMPOSITION TO SUPPORT COMPLEX AND TAILORABLE PROCESS DEFINITIONS

SEOK-WON LEE, ROBIN A. GANDHI, SIDDHARTH J. WAGLE

Knowledge-intensive Software Engineering (NiSE) Research Group
Dept. of Software and Information Systems, The University of North Carolina at Charlotte
Charlotte, NC 28223-0001, USA.
{seoklee, rgandhi, sjwagle}@uncc.edu
http://nise.sis.uncc.edu

Received (7 March 2008)
Revised (Day Month Year)
Accepted (Day Month Year)

Services as abstractions of functionality have enabled the engineering of systems that support well-defined processes with relative ease. This success leads to aspirations for achieving greater complexity with the service-oriented paradigm. In particular, we address the case where the process definition is tailored differently in each instantiation based on negotiations among stakeholders of a socio-technical context. For such cases the process definition invariably crosscuts the architecture of a process-support system that composes available services. However, use of pre-defined process variations may bias the tailoring effort and thus, act against the original motivation of having a flexible definition. On the other hand, the characteristics of process complexity and tailorability introduce differences between stakeholder understanding of the process activities and their manifestation in tool support. We encounter these issues while developing a service-oriented process-support system for a security Certification and Accreditation (C&A) process. In this paper, we present our approach to effectively separate the C&A process definition from the architecture of its process-support system. We employ ontological modeling techniques to explicitly model the process definition and later expose it as a service to provide weaving rules for dynamically composing the process-support system architecture at runtime. The feasibility of our approach has been demonstrated in the design of a service-oriented architecture for a prototype workbench that supports the Department of Defense Certification and Accreditation Process (DITSCAP).

Keywords: Service-oriented architecture, Aspect-oriented design, Ontology-based domain modeling, Dynamic architecture composition, Model-driven engineering, Certification and Accreditation.

1. Introduction

Service Oriented Architecture (SOA) is enabled through an interconnected set of services, each accessible through standard interfaces and messaging protocols [31] [37]. S*ervices* as first class entities offer functional abstractions that are extensible, loosely-coupled, and reusable. These characteristics of services drive the vision of a flexible and distributed infrastructure that supports on-demand business needs. For example, using web services [17] abstract process workflows can be built by simply orchestrating interactions among several distributed services over the Internet using specifications such as the Business Process Execution Language (BPEL) [25]. However, this architectural

Accepted on July 14, 2008

style assumes the existence of an abstract and well-defined process workflow model and ignores the reality of organizational and human influence on the definition and execution of a process [10]. These influences induce a complex flow of artifacts between social and technical worlds separated at the boundary of a designed process-support system. Furthermore, a situation commonly arises where the process definition is tailored differently in each of its instantiation based on the negotiations among stakeholders in a socio-technical context. For such cases the process definition invariably crosscuts the architecture of a process-support system that composes available services. However, use of pre-defined process variations to address these issues may bias the tailoring effort and thus, act against the original motivation of having a flexible definition. Therefore, we outline important requirements for a SOA to support a complex and tailorable process:

- The SOA should be composed in accordance with the tailoring effort of the process definition
- The SOA should maintain a chain of evidence for process fulfillment in a socio-technical environment where complete automation is not desired
- The SOA should facilitate stakeholder understanding of the process definition and its realizations through the process-support system

To fulfill these key requirements, the process definition that drives the composition of services in a SOA needs to be "framed" and separated based on a knowledge-intensive approach [6]. This approach emphasizes "*services*" based on deep representation of the process definition itself that provide intelligent assistance to understand the coordinated interactions between stakeholders and the process-support system, as well as the guidance to the composition of the required architecture. Based on this philosophy, in this paper we outline our approach to design a service-oriented process-support system for a complex and tailorable Certification and Accreditation (C&A) process.

Security C&A process is defined as the comprehensive evaluation of the technical and non-technical security features of a software system to establish the extent to which a particular design and implementation meets a set of specified regulatory requirements [14]. However, the resources required for understanding the C&A process and resources for carrying out its activities are usually scattered in multiple documents/sources at different levels in the organizational hierarchy. Therefore, effective execution of the C&A process demands a unified access and view to common resources such as the organizational policies, certification requirements, system-security information, and other artifacts. To this end, *services* provide highly distributed and reusable ways to aggregate, abstract and disseminate access to these common resources in the design of a process-support system for C&A (Section 4).

C&A is a long and exhaustive process based on a set of activities defined by regulations [27]. Infrastructure-wide C&A processes usually recommend a risk based approach to come up cost-effective security solutions in the context of a particular software system. Therefore, tailorability of the C&A process is fundamental for its applicability into the developmental and operational processes of diverse software systems deployed in the organizational infrastructure. The C&A process is designed to be

Accepted on July 14, 2008

tailored such that it can be practiced for any system regardless of the system status in its life cycle (inception, development, deployed, etc.) or shift in program strategy (grand design, incremental, or evolutionary) [13]. C&A process evolution/improvement is also continuously motivated by factors such as the ever increasing complexities of organizational software systems; changes in the perceived types and levels of threats; or as the applicable threats change over time. In addition, several organizational and human aspects are involved in engineering the C&A process execution that fits the needs of a particular software system and its operational environment. The key roles involved in negotiating the C&A process definition are representative participants from the diverse areas in the organization. The characteristics motivate the design of the architecture for a C&A process-support system that effectively addresses the complexity as well as the need for tailorability of the process definition.

To address these needs, we combine techniques in ontology-based domain modeling [54] [43] and aspect-oriented design [18] [40] [36] to modularize the process definition as a human and machine understandable representation in a larger service-oriented design solution. Specifically, we capture the process definition as an ontological model, called the "*process ontology*" (Section 5). The purpose of the process ontology is to maintain explicit traceability between the purpose of C&A activities and the available software artifacts (e.g. services, user interface components, etc.) of a process-support system. To achieve this, the process ontology is a hierarchical decomposition of high level strategic C&A goals into specific tasks supported by the process-support system. Each task in the process ontology is associated with "*architectural weaving rules*" that specify *what* service compositions are necessary in the process-support system architecture *when* certain tasks are encountered during the process execution (Section 5.2). To demonstrate the feasibility of our approach, we provide examples from the design of the architecture of a prototype workbench [46] that supports the Department of Defense security C&A Process (DITSCAP) [14].

The rest of the paper is organized as follows. Section 2 provides background on the relevant aspects of our previous research while motivating the use of services to support the C&A process. In section 3, we provide a conceptual overview of a service-oriented workbench architecture for supporting the C&A process. In section 4, we elaborate on the development of distributed and reusable service definitions to support C&A process tasks. Section 5 outlines the methodological steps involved in the development of a C&A process ontology, followed by the demonstration of its usage for architectural composition of the workbench in section 6. Section 7 presents some related work followed by contributions and future work in section 8.

2. Background and Motivation

C&A processes assess the level of compliance of a software system against a set of baseline security requirements. These requirements are usually scattered across many natural language regulatory documents and their compliance evidences are gathered from heterogeneous sources based on domain expertise of those conducting the C&A process.

Accepted on July 14, 2008

Consequently, C&A processes often lack consistent and comparable results and fail to provide adequate information for authorizing officials to understand security risks and make informed decisions [51] [33]. To address these issues, we discuss our previous research efforts towards modeling regulatory security requirements. Our techniques [43] [41] have been applied to model the requirements for the DITSCAP [14] with promising preliminary results [38] [42] [44] [45].

2.1. *Regulatory Security Requirements Modeling*

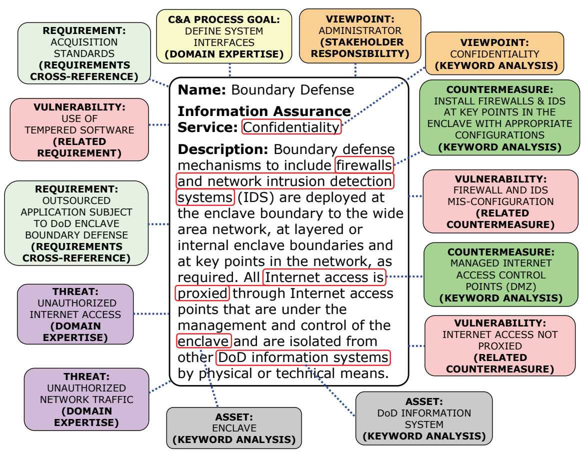
Our goal to model regulatory security requirements is to facilitate a common understanding of the complex constraints imposed by them on software behavior in a socio-technical environment. Therefore, we have applied the Ontology-based ACTive Requirements Engineering (Onto-ActRE) framework [43] for modeling and analyzing requirements specified in regulatory documents by utilizing the synergy among multiple requirements modeling philosophies. The Onto-ActRE approach to ontology development is primarily problem driven; i.e. ontology development is guided based on the problem solving notions of goals, scenarios, and viewpoints (requirements engineering techniques). Driven by these modeling philosophies, we extract ontological concepts from natural language regulatory documents as well as domain experts to help in classifying and categorizing regulatory security requirements from multiple dimensions [44]. The result of applying the Onto-ActRE framework is a Problem Domain Ontology (PDO), which includes the followings: 1) a hierarchical requirements domain model of various requirement types that categorize regulatory security requirements; 2) a viewpoints hierarchy that models different perspectives from related stakeholders of a regulatory security requirement; 3) a C&A process goal hierarchy with leaf-node scenarios to gather user/system criteria for regulatory security requirements applicability; 4) domain specific taxonomies with ontological concepts in the dimensions of threats, countermeasures, vulnerabilities, and assets related to regulatory security requirements for understanding risks associated with non-compliance; and 5) interdependencies among the concepts modeled in the PDO. Further details about these models are described in [44] [45] [48]. Semantics of a requirement is now reflected by its relationships with other concepts in the PDO. As an example, Fig. 1 shows the DITSCAP requirement of "Boundary Defense" [15], its related domain concepts and their methods of identification.

Support for ontological domain modeling for the Onto-ActRE framework is provided by the GENeric Object Model (GenOM) [41] toolkit. GenOM inherits the theoretical foundation of the frame representation and is compatible with the OKBC specification [52] for knowledge representation and sharing.

2.2. *What services are necessary for a C&A process-support system?*

Regulatory guidance documents (e.g. the DITSCAP Application Manual [13]) specify the C&A process at an abstract level to maintain general applicability across diverse

Accepted on July 14, 2008


Name: Boundary Defense

Information Assurance Service: Confidentiality

Description: Boundary defense mechanisms to include firewalls and network intrusion detection systems (IDS) are deployed at the enclave boundary to the wide area network, at layered or internal enclave boundaries and at key points in the network, as required. All Internet access is proxied through Internet access points that are under the management and control of the enclave and are isolated from other DoD information systems by physical or technical means.

REQUIREMENT: ACQUISITION STANDARDS (REQUIREMENTS CROSS-REFERENCE)
C&A PROCESS GOAL: DEFINE SYSTEM INTERFACES (DOMAIN EXPERTISE)
VIEWPOINT: ADMINISTRATOR (STAKEHOLDER RESPONSIBILITY)
VIEWPOINT: CONFIDENTIALITY (KEYWORD ANALYSIS)
VULNERABILITY: USE OF TEMPERED SOFTWARE (RELATED REQUIREMENT)
COUNTERMEASURE: INSTALL FIREWALLS & IDS AT KEY POINTS IN THE ENCLAVE WITH APPROPRIATE CONFIGURATIONS (KEYWORD ANALYSIS)
REQUIREMENT: OUTSOURCED APPLICATION SUBJECT TO DoD ENCLAVE BOUNDARY DEFENSE (REQUIREMENTS CROSS-REFERENCE)
VULNERABILITY: FIREWALL AND IDS MIS-CONFIGURATION (RELATED COUNTERMEASURE)
COUNTERMEASURE: MANAGED INTERNET ACCESS CONTROL POINTS (DMZ) (KEYWORD ANALYSIS)
THREAT: UNAUTHORIZED INTERNET ACCESS (DOMAIN EXPERTISE)
VULNERABILITY: INTERNET ACCESS NOT PROXIED (RELATED COUNTERMEASURE)
THREAT: UNAUTHORIZED NETWORK TRAFFIC (DOMAIN EXPERTISE)
ASSET: ENCLAVE (KEYWORD ANALYSIS)
ASSET: DoD INFORMATION SYSTEM (KEYWORD ANALYSIS)

Fig. 1. A DITSCAP requirement and its relationships with other domain concepts in the PDO [38]

software systems in an organizational infrastructure. However, such abstract specification leaves the C&A process definition open to subjective interpretation. Stakeholders often find the C&A process hard to understand and trace its implementations in practice back to high level strategic goals or justify its rational and repeatability. The complexity of the C&A process and its tailorability further raise concerns for the gap between stakeholders' understanding of the process definition. Therefore, it is important for any C&A process-support system to promote stakeholders' awareness about the process activities that are partially or fully supported. Fulfilling these needs in a SOA emphasizes the need for *services* that are based on deep representation of the C&A process definition itself. Such a service will help to maintain a chain of evidence for process fulfillment in a socio-technical environment where complete automation is neither desired nor possible.

In addition to process guidance, the large spread of C&A activities requires services that provide a uniform access and view to compliance evidences from heterogeneous sources (e.g. domain experts, task reports, software assurance tools, etc). Design of services that facilitate the aggregation, recall and analysis of artifacts from data sources that cannot be anticipated in advance are necessary for an effective and scalable C&A process-support system. In the following section, the conceptual architecture of a C&A process-support system based on services is presented.

3. A Conceptual C&A Workbench Architecture based on Services

Currently, the notion of services is limited to provide functional abstractions. The process definition then provides guidance on how to compose the available services. However, the process itself has not been captured as a "service" due to its lack of functional characteristics. Consequently, process definitions do not use the same infrastructure and frameworks that have been built to design and support flexible services. It is not surprising that current SOA implementations require additional languages to model and

Accepted on July 14, 2008

interpret process definitions that compose available services and orchestrate their interactions. In this paper, we describe a knowledge-intensive approach to expose the process definition itself as a service (Section 5). A "process service" provides abstractions of the process definition that guides the composition of other functional services in a SOA. In effect, the process service, a deep knowledge representation of the process itself, modularizes the process related cross-cutting concerns that are scattered throughout the SOA. Based on the aspect-oriented design paradigm, such modularized cross-cutting concerns are called *aspects*. Hence, we further qualify the process service as a "*process-aspect knowledge service*".

With the existence of a process-aspect knowledge service, a SOA is able to maintain a clear separation between: 1) the data and control; and 2) the static and dynamic software artifacts in its architecture. A conceptual overview of the information flow that leads to the C&A process-support workbench based on such an architectural style is shown in Fig. 2.

The stakeholders practicing the C&A process (e.g. the certifier, certification team, user representative, etc.) actively influence the definition and design of the software artifacts that provide access to the heterogeneous data sources required for conducting C&A tasks. These software artifacts support the collection, organization, retrieval and analysis of compliance evidences gathered from the software system being certified. The software artifacts include the designed services that provide access to compliance evidences based on a rich classification and categorization of domain concepts in the PDO. As a result, we further qualify these services as "*knowledge services*" in Fig. 2. The knowledge services (Section 4) are designed to be consumed by "*process-support*

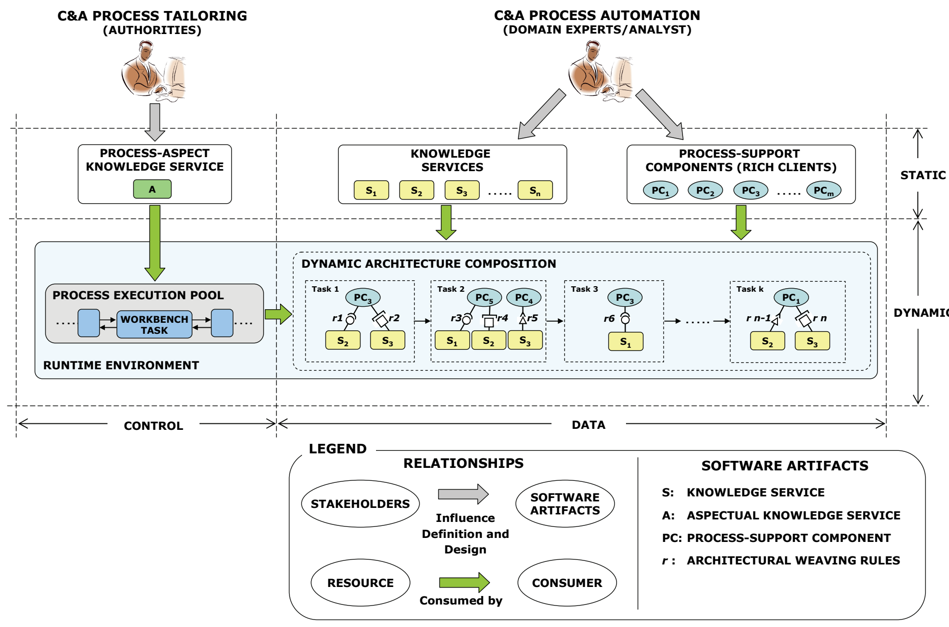

C&A PROCESS TAILORING (AUTHORITIES)
C&A PROCESS AUTOMATION (DOMAIN EXPERTS/ANALYST)
PROCESS-ASPECT KNOWLEDGE SERVICE
A
KNOWLEDGE SERVICES
S1 S2 S3 ..... Sn
PROCESS-SUPPORT COMPONENTS (RICH CLIENTS)
PC1 PC2 PC3 ..... PCm
STATIC
DYNAMIC
DYNAMIC ARCHITECTURE COMPOSITION
PROCESS EXECUTION POOL
WORKBENCH TASK
RUNTIME ENVIRONMENT
Task 1
Task 2
Task 3
Task k
r1 r2 r3 r4 r5 r6 r n-1 r n
CONTROL
DATA
LEGEND
RELATIONSHIPS
STAKEHOLDERS
SOFTWARE ARTIFACTS
Influence Definition and Design
RESOURCE
CONSUMER
Consumed by
SOFTWARE ARTIFACTS
S: KNOWLEDGE SERVICE
A: ASPECTUAL KNOWLEDGE SERVICE
PC: PROCESS-SUPPORT COMPONENT
r: ARCHITECTURAL WEAVING RULES

Fig. 2. Conceptual Overview of the C&A Workbench Architecture

Accepted on July 14, 2008

components" (Section 5.2.1), which are software artifacts that allow several analytical operations to be performed on the compliance evidences.

Stakeholders who negotiate the C&A process definition based on the needs of a particular system and organization actively influence the definition and design of the software artifacts that provide control information for process execution. To this end, the authorized C&A officials (e.g. the Designated Approving Authority (DAA) and the acquisition organization Program Manager) consider various factors such as the mission criticality, software system lifecycle strategy, and many others to negotiate the C&A process definition and the level of effort required. These negotiations help identify the high-level C&A goals which are eventually satisfied by tasks supported through the workbench. The construction of a process ontology captures this knowledge in a hierarchical manner using ontological domain modeling techniques (Section 5). The process ontology associates specific tasks with "*architectural weaving rules*" (Section 5.2.2) that compose knowledge services for consumption by process-support components during process execution. To support such workbench architecture configuration at runtime, the process ontology and related architectural weaving rules are exposed through operations defined by the process-aspect knowledge service (Section 6), shown in Fig. 2.

The workbench architecture can also be logically separated into static and dynamic software artifacts. The *static* software artifacts of the workbench include the *knowledge services* (data); highly modularized and reusable process-support components (to access and analyze data); and the process-aspect knowledge service (control). The *dynamic* software artifacts are created at runtime using a composition algorithm (Section 6.1) by selection and activation of appropriate static software artifacts.

The C&A workbench architecture is in essence composed based on the representation of domain concepts from the human and machine understandable ontologies in the PDO, whose structure and content is influenced by multiple stakeholders and regulatory texts. The contents of the compliance evidences (instance space) gathered might change; but, the domain concepts in the ontology (conceptual space) that classify and categorize them are relatively stable. Therefore, the availability of evidences from heterogeneous and unknown sources does not warrant a change in the services that provide access to them. In the following sections, we discuss parts of the conceptual workbench architecture in further detail.

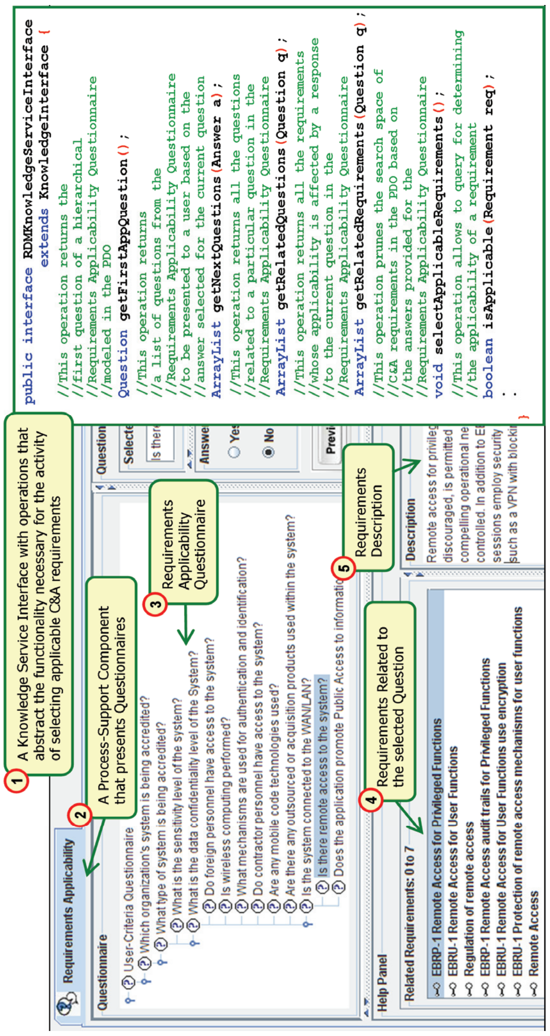
4. Use of *Services* to encapsulate Knowledge Models

The domain concepts modeled in the PDO provide placeholders to gather information regarding the software system from user representatives, certification analysts, operating manuals, plans, architecture diagrams, automated network-based information discovery toolkits and many other sources [44]. Therefore, to accomplish the tasks in a C&A process, the PDO as an exhaustive classification and categorization of C&A requirements is a good candidate to be exposed through *services.* These services will allow flexible aggregation of data from heterogeneous sources and then help to disseminate it through process-support components that interface with domain experts.

Accepted on July 14, 2008

4.1. *Defining Knowledge Services for the C&A process*

From experiences in the web services domain, it has been observed that services exposed as a collection of well-defined functionality are more flexible in terms of their usage by other services or tools [1]. Similarly, exposing the PDO knowledge models through *services* involves determining well-defined collections of functionalities to support knowledge-intensive C&A tasks. The role of these services is to aggregate the fundamental methods provided by a knowledge base such as edit, browse, access, query, infer, and visualize ontological models to build richer functional abstractions that are relevant while performing the C&A tasks. We refer to these functional abstractions as


1 A Knowledge Service Interface with operations that abstract the functionality necessary for the activity of selecting applicable C&A requirements
2 A Process-Support Component that presents Questionnaires
3 Requirements Applicability Questionnaire
4 Requirements Related to the selected Question
5 Requirements Description
Requirements Applicability
Questionnaire
Question
Selecte
Answe
Yes
No
Previ
Help Panel
Related Requirements: 0 to 7
Description

User-Criteria Questionnaire
Which organization's system is being accredited?
What type of system is being accredited?
What is the sensitivity level of the system?
What is the data confidentiality level of the System?
Do foreign personnel have access to the system?
Is wireless computing performed?
What mechanisms are used for authentication and identification?
Do contractor personnel have access to the system?
Are any mobile code technologies used?
Are there any outsourced or acquisition products used within the system?
Is the system connected to the WAN/LAN?
Is there remote access to the system?
Does the application promote Public Access to informatio

EBRP-1 Remote Access for Privileged Functions
EBRU-1 Remote Access for User Functions
Regulation of remote access
EBRP-1 Remote Access audit trails for Privileged Functions
EBRU-1 Remote Access for User Functions use encryption
EBRU-1 Protection of remote access mechanisms for user functions
Remote Access

Remote access for privileg
discouraged, is permitted
compelling operational ne
controlled. In addition to E
sessions employ security
such as a VPN with blocki

```
public interface RDMKnowledgeServiceInterface
                  extends KnowledgeInterface {
  //This operation returns the
  //first question of a hierarchical
  //Requirements Applicability Questionnaire
  //modeled in the PDO
  Question getFirstAppQuestion();
  //This operation returns
  //a list of questions from the
  //Requirements Applicability Questionnaire
  //to be presented to a user based on the
  //answer selected for the current question
  ArrayList getNextQuestions(Answer a);
  //This operation returns all the questions
  //related to a particular question in the
  //Requirements Applicability Questionnaire
  ArrayList getRelatedQuestions(Question q);
  //This operation returns all the requirements
  //whose applicability is affected by a response
  //to the current question in the
  //Requirements Applicability Questionnaire
  ArrayList getRelatedRequirements(Question q);
  //This operation prunes the search space of
  //C&A requirements in the PDO based on
  //the answers provided for the
  //Requirements Applicability Questionnaire
  void selectApplicableRequirements();
  //This operation allows to query for determining
  //the applicability of a requirement
  boolean isApplicable(Requirement req);
  :
}
```

Fig. 3. A Process-support Component in the C&A Workbench and the Requirements Domain Model (RDM) Knowledge Service Interface written in Java to support the DITSCAP activity of determining applicable C&A requirements for a particular software system.

Accepted on July 14, 2008

Knowledge Services. The knowledge services are eventually consumed by process-support components. The process-support components are rich clients that engage a certification analyst or other stakeholders in interactive analytical sessions to produce C&A artifacts or analyze evidences gathered from the target system.

As an example, consider the task of "select applicable C&A requirements" for a system subject to the DITSCAP. Following a manual approach, a certification analyst is expected to sift through numerous regulatory documents evaluating the applicability of requirements specified in natural language. In contrast, the Requirements Domain Model (RDM) of the PDO provides a rich classification and categorization of regulatory requirements in the DITSCAP domain. The RDM also includes a well-designed requirements applicability questionnaire [44] whose answer options prune the requirements search space to determine the set of requirements applicable to a particular system. Now, to effectively support the task of "select applicable C&A requirements," the fundamental knowledge base operations upon the RDM are combined to provide functional abstractions that are exposed as a *knowledge service*. As an example, Fig. 3 Label 1 shows a partial RDM knowledge service interface with operations that abstract the functionality necessary for the task of "select applicable C&A requirements". Fig. 3 Labels 2 through 5 depict the process-support components that use these functional abstractions to present requirements applicability questionnaires to a certification analyst.

Other knowledge services in the DITSCAP domain expose the classification and categorization of the following concepts in the PDO associated with security requirements: 1) viewpoints; 2) C&A process goals; and 3) various risks components (threats, assets, countermeasures, vulnerabilities). Fig. 4 provides a conceptual overview of these knowledge services defined to expose the models of the PDO. The knowledge services build rich functional abstractions upon the OKBC [52] compliant APIs supported by our knowledge base (GenOM).

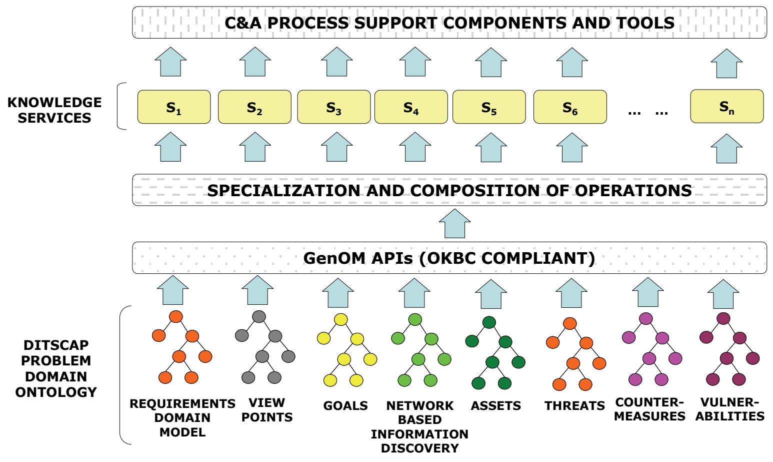

C&A PROCESS SUPPORT COMPONENTS AND TOOLS
KNOWLEDGE SERVICES
S1
S2
S3
S4
S5
S6
... ...
Sn
SPECIALIZATION AND COMPOSITION OF OPERATIONS
GenOM APIs (OKBC COMPLIANT)
DITSCAP PROBLEM DOMAIN ONTOLOGY
REQUIREMENTS DOMAIN MODEL
VIEW POINTS
GOALS
NETWORK BASED INFORMATION DISCOVERY
ASSETS
THREATS
COUNTER-MEASURES
VULNER-ABILITIES

Fig. 4. Knowledge Services defined to support the DITSCAP

Accepted on July 14, 2008

4.1.1. *The Knowledge Service Interface definition*

A *service interface* represents a contract/agreement that sets the expectations for the collaborating entities [37]. Furthermore, for simplified usage, the service interface should represent well-defined and abstract collection of functionality that resembles the logical units of operations required to execute process tasks. However, a balance should be maintained between achieving such abstraction and allowing flexibility for future extension and evolution of services. To address this issue, we perform an incremental aggregation of the operations supported by the knowledge base APIs at different levels of granularity while constructing the definition of a knowledge service interface. A gradual and layered aggregation of operations allows the right level of abstraction to be determined rationally rather than relying on subjective intuition.

As an example of this design philosophy, consider the most fine-grained operations that are provided by the knowledge base APIs (OKBC compliant) to edit, browse, access, query, infer, and visualize ontological models as shown in Fig. 5.
At the next level of granularity, atomic operations specialize generic operations based on the domain concepts defined in the PDO. For example, to navigate a hierarchical collection of categories in the PDO requires the definition of atomic operations such as "*get all categories that are subclasses of a given category"*. In the next level of abstraction, the atomic operations are composed into higher-level *composite operations* with additional business logic (as glue) for providing functionality required to achieve process-dependent tasks. For example, "*select applicable requirements*" operation is an aggregate of several atomic operations to select the requirements applicable to the target system based on the answers to requirements applicability questions during the C&A process. Finally, the knowledge service interface is built by a selective aggregation of composite as well as atomic operations such that intuitive logical units of operations are exposed through the knowledge service. Such incremental aggregation of functionality ensures a combined impact of the domain concepts in the PDO as well as process activities on the definition of the knowledge service interface. A partial knowledge service interface definition is shown in Fig. 3 (Label 1).

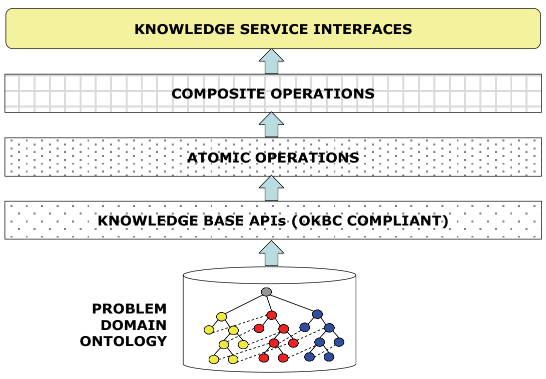

KNOWLEDGE SERVICE INTERFACES
COMPOSITE OPERATIONS
ATOMIC OPERATIONS
KNOWLEDGE BASE APIs (OKBC COMPLIANT)
PROBLEM DOMAIN ONTOLOGY

Fig. 5. Aggregation of Functionalities for Knowledge Service Interface definition

Accepted on July 14, 2008

4.1.2. *Deploying, Discovering, and Invoking Knowledge Services*

In the DITSCAP domain, *integrity* is an important QoS factor for knowledge services that support C&A activities of a critical software system. We define integrity of knowledge services as the accurate and orderly delivery of messages to service consumers, i.e., the components of the workbench which facilitate decision making of stakeholders in the C&A process. To address these integrity requirements we apply appropriate design patterns while deploying knowledge services for the C&A workbench. Specifically, for the deployment of knowledge services, we use the *Factory* and *Singleton* design patterns [16] that are at the core of several SOA technologies.

As shown in Fig. 6, we define a "Knowledge Factory" class that aggregates various knowledge service interfaces and provides reference to a *singleton* instance of the knowledge service implementation to a consumer/requestor. Essentially, the "Knowledge Factory" class is a means to aggregate concrete singleton knowledge service endpoints while providing the service consumers with a common gateway to leverage the functionality offered by the knowledge services. The "Knowledge Factory" class parameterizes each service endpoint with the service name/URI to make them discoverable at runtime.

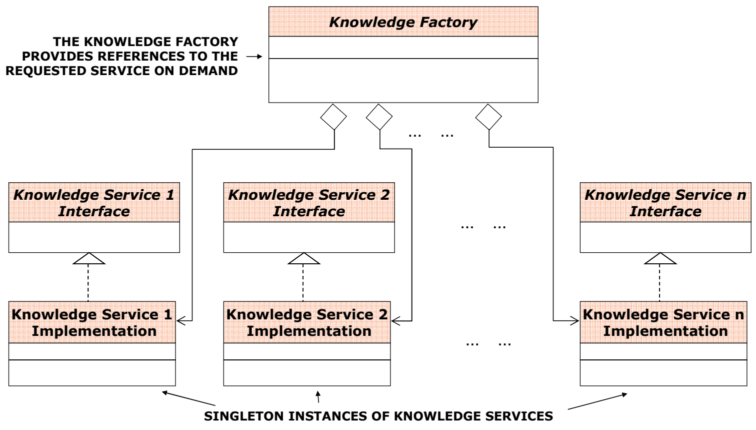

Knowledge Factory
THE KNOWLEDGE FACTORY PROVIDES REFERENCES TO THE REQUESTED SERVICE ON DEMAND
Knowledge Service 1 Interface
Knowledge Service 2 Interface
Knowledge Service n Interface
Knowledge Service 1 Implementation
Knowledge Service 2 Implementation
Knowledge Service n Implementation
SINGLETON INSTANCES OF KNOWLEDGE SERVICES

Fig. 6. Use of Factory and Singleton Design Patterns to Deploy, Discover and Invoke Knowledge Services

5. Modeling the Process Ontology

In this section, we discuss the methodological steps to build a process ontology for a C&A process starting from its specification in regulatory documents. We use ontology development as a way to capture the rationale of the process and its tailoring effort for a particular instantiation. For the DITSCAP, the process definition is an abstract specification of multiple interconnected *workflows* required to produce artifacts that satisfy the strategic C&A goals for the target system. A workflow is an ordered collection of related C&A *activities*. Each activity in a workflow can be further decomposed into *atomic tasks* that are carried out in practice. Although well-defined from the business

Accepted on July 14, 2008

mission aspect, the abstract C&A process model can be instantiated in many ways to fit the characteristics of the system being certified [13]. To this end, our approach serves as a way to systematically understand and establish an explicit agreement of the process definition among stakeholders.

5.1. *Building an Ontological Process Model*

5.1.1. *Operationalization of the Process Goals*

To create explicit traceability between the strategic C&A process goals and the tasks that satisfy the goals in practice, we build the process ontology following a hierarchical goal decomposition approach. Specifically, we adopt the goal decomposition technique in requirements engineering [53] to operationalize the high level process goals into specific process activities at different levels of abstraction. We use DITSCAP to demonstrate our approach while suggesting potential sources for process ontology construction. As an example, Fig. 7 shows a partial process ontology of the DITSCAP and its concepts at different levels of abstractions extracted from regulatory documents.

The DITSCAP application manual [13] is a comprehensive and authoritative document which provides implementation guidance to standardize the C&A process throughout the United States Department of Defense (DoD). Based on this document, the high level goals for DITSCAP are to generate a comprehensive system definition (security plan); perform risk assessment; and maintain operational system security. These goals can be further decomposed into more specific goals as prescribed by the process workflows in the DITSCAP application manual [13]. For example, the DITSCAP "Perform System Registration" goal is operationalized by a workflow, which is a grouping of activities that *starts* with the "Mission/System Description" activity and is *finalized* by the "Draft SSAA" activity, as shown in Fig. 7 (Level 2).

At the next level of abstraction, each process activity is further operationalized by *atomic tasks*. An atomic task cannot be operationlized further and it needs to be executed either manually or through automated tool support in the workbench. Fig. 7 (Level 3) shows the atomic tasks that operationalize the activities defined in the previous level.

5.1.2. *Understanding Process Automation*

A process-support system has to support interactivity and co-operation between automated and manual tasks in a socio-technical environment. The automated tasks may differ in granularity and sequence compared to their manual counterparts. Therefore, to provide a clear understanding of the process automation areas as well as the interdependencies with other manual activities, the tasks automated by the workbench should be explicitly traceable to the atomic tasks (Fig. 7 (Level 3)) to which they contribute. Such traceability provides justification of adhering to the process definition and increases awareness of high-level process goals while conducting specific tasks [49]. It also facilitates later interpretation and re-composition of gathered process artifacts.

Accepted on July 14, 2008

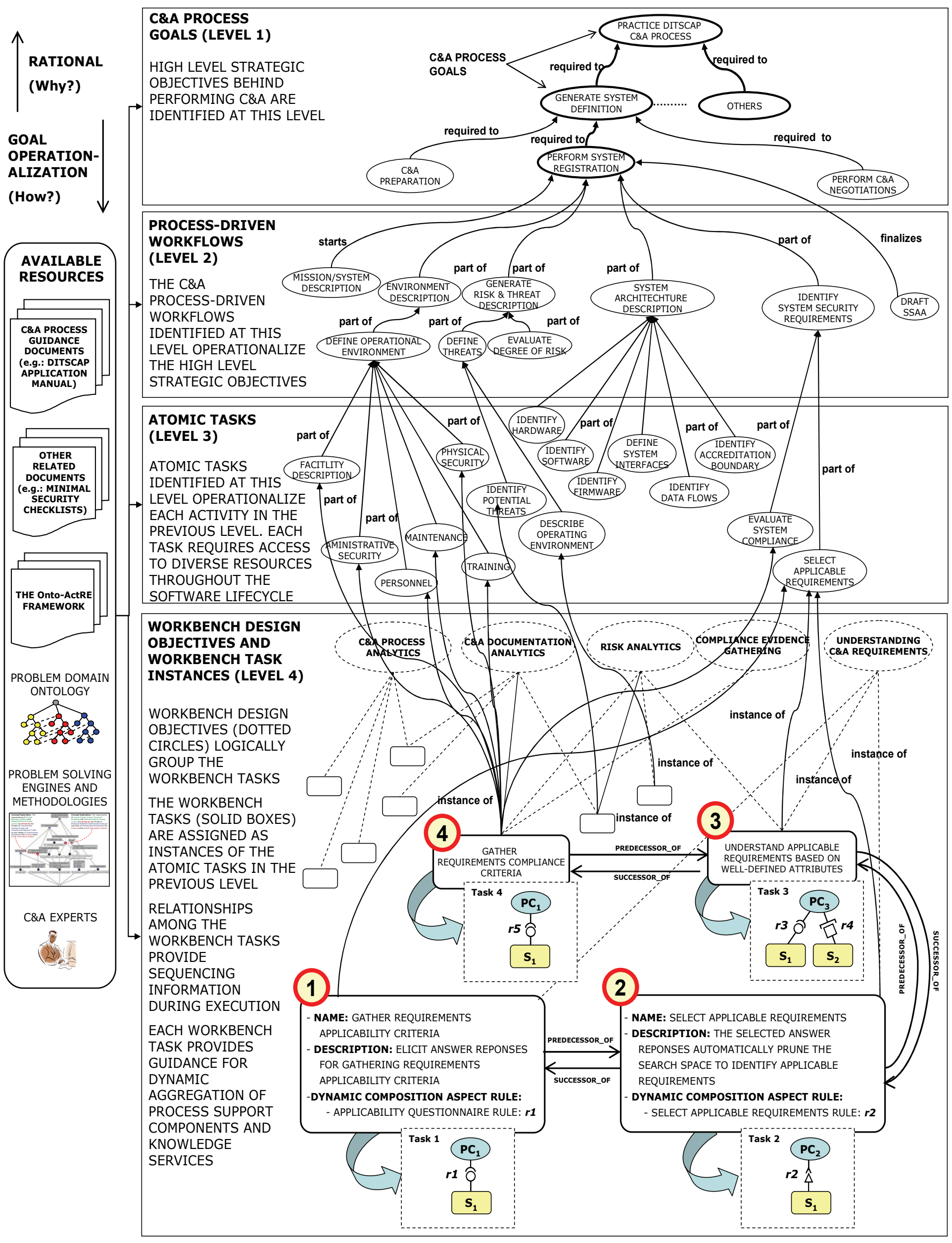

RATIONAL (Why?)
GOAL OPERATION-ALIZATION (How?)
AVAILABLE RESOURCES
C&A PROCESS GUIDANCE DOCUMENTS (e.g.: DITSCAP APPLICATION MANUAL)
OTHER RELATED DOCUMENTS (e.g.: MINIMAL SECURITY CHECKLISTS)
THE Onto-ActRE FRAMEWORK
PROBLEM DOMAIN ONTOLOGY
PROBLEM SOLVING ENGINES AND METHODOLOGIES
C&A EXPERTS
C&A PROCESS GOALS (LEVEL 1)
HIGH LEVEL STRATEGIC OBJECTIVES BEHIND PERFORMING C&A ARE IDENTIFIED AT THIS LEVEL
C&A PROCESS GOALS
PRACTICE DITSCAP C&A PROCESS
required to
required to
GENERATE SYSTEM DEFINITION
OTHERS
required to
required to
required to
C&A PREPARATION
PERFORM SYSTEM REGISTRATION
PERFORM C&A NEGOTIATIONS
PROCESS-DRIVEN WORKFLOWS (LEVEL 2)
THE C&A PROCESS-DRIVEN WORKFLOWS IDENTIFIED AT THIS LEVEL OPERATIONALIZE THE HIGH LEVEL STRATEGIC OBJECTIVES
starts
part of
part of
part of
part of
finalizes
MISSION/SYSTEM DESCRIPTION
ENVIRONMENT DESCRIPTION
GENERATE RISK & THREAT DESCRIPTION
SYSTEM ARCHITECHTURE DESCRIPTION
IDENTIFY SYSTEM SECURITY REQUIREMENTS
DRAFT SSAA
part of
part of
part of
DEFINE OPERATIONAL ENVIRONMENT
DEFINE THREATS
EVALUATE DEGREE OF RISK
ATOMIC TASKS (LEVEL 3)
ATOMIC TASKS IDENTIFIED AT THIS LEVEL OPERATIONALIZE EACH ACTIVITY IN THE PREVIOUS LEVEL. EACH TASK REQUIRES ACCESS TO DIVERSE RESOURCES THROUGHOUT THE SOFTWARE LIFECYCLE
part of
part of
part of
part of
part of
part of
part of
FACILITY DESCRIPTION
PHYSICAL SECURITY
IDENTIFY HARDWARE
IDENTIFY SOFTWARE
DEFINE SYSTEM INTERFACES
IDENTIFY ACCREDITATION BOUNDARY
IDENTIFY FIRMWARE
IDENTIFY DATA FLOWS
IDENTIFY POTENTIAL THREATS
DESCRIBE OPERATING ENVIRONMENT
EVALUATE SYSTEM COMPLIANCE
SELECT APPLICABLE REQUIREMENTS
AMINISTRATIVE SECURITY
MAINTENANCE
TRAINING
PERSONNEL
WORKBENCH DESIGN OBJECTIVES AND WORKBENCH TASK INSTANCES (LEVEL 4)
WORKBENCH DESIGN OBJECTIVES (DOTTED CIRCLES) LOGICALLY GROUP THE WORKBENCH TASKS
THE WORKBENCH TASKS (SOLID BOXES) ARE ASSIGNED AS INSTANCES OF THE ATOMIC TASKS IN THE PREVIOUS LEVEL
RELATIONSHIPS AMONG THE WORKBENCH TASKS PROVIDE SEQUENCING INFORMATION DURING EXECUTION
EACH WORKBENCH TASK PROVIDES GUIDANCE FOR DYNAMIC AGGREGATION OF PROCESS SUPPORT COMPONENTS AND KNOWLEDGE SERVICES
C&A PROCESS ANALYTICS
C&A DOCUMENTATION ANALYTICS
RISK ANALYTICS
COMPLIANCE EVIDENCE GATHERING
UNDERSTANDING C&A REQUIREMENTS
instance of
instance of
instance of
instance of
instance of
instance of
4
GATHER REQUIREMENTS COMPLIANCE CRITERIA
3
UNDERSTAND APPLICABLE REQUIREMENTS BASED ON WELL-DEFINED ATTRIBUTES
PREDECESSOR_OF
SUCCESSOR_OF
Task 4
PC_1
r5
S_1
Task 3
PC_3
r3
r4
S_1
S_2
PREDECESSOR_OF
SUCCESSOR_OF
1
- NAME: GATHER REQUIREMENTS APPLICABILITY CRITERIA
- DESCRIPTION: ELICIT ANSWER REPONSES FOR GATHERING REQUIREMENTS APPLICABILITY CRITERIA
-DYNAMIC COMPOSITION ASPECT RULE:
- APPLICABILITY QUESTIONNAIRE RULE: r1
PREDECESSOR_OF
SUCCESSOR_OF
2
- NAME: SELECT APPLICABLE REQUIREMENTS
- DESCRIPTION: THE SELECTED ANSWER REPONSES AUTOMATICALLY PRUNE THE SEARCH SPACE TO IDENTIFY APPLICABLE REQUIREMENTS
- DYNAMIC COMPOSITION ASPECT RULE:
- SELECT APPLICABLE REQUIREMENTS RULE: r2
Task 1
PC_1
r1
S_1
Task 2
PC_2
r2
S_1

Fig. 7. Modeling Workbench Tasks as Instances of Process Workflow Tasks

To ease the process of achieving end-to-end traceability, the automation areas of the C&A workbench are logically divided into *workbench design objectives*. The workbench design objectives group *workbench tasks*, which in turn contribute to accomplish the atomic tasks. Fig. 7 (Level 4) provides a high level overview of this modeling activity.

Accepted on July 14, 2008

Many-to-many mappings between the workbench tasks instances and the atomic tasks are the result of differences in the level of granularity and/or sequence among them. For example in Fig. 7, the workbench task labeled (4 in Level 4) contributes to several atomic tasks in Level 3; whereas, the tasks labeled (1, 2 and 3 in Level 4) all contribute only to single "Select Applicable Requirements" atomic task in Level 3.

The workbench tasks maintain explicit interdependencies among them as predecessors and successors of each other to provide sequencing mechanism during user interaction with the workbench. For example in Fig. 7, the task "Gather Requirements Applicability Criteria" (Label 1) is a predecessor of the task "Select Applicable Requirements" (Label 2), to first gather the requirements applicability criteria from the certification analyst before choosing the applicable requirements.

5.2. *Modeling a Workbench Task*

The process ontology development until now focused on the decomposition of high-level process goals to yield a set of workbench tasks for automation. This conceptual decomposition has also driven the effort to tailor the generic C&A process based on factors such as the mission criticality; software system lifecycle strategy (waterfall, spiral, etc.); or the stage of the software system lifecycle in which the certification activities are initiated. The next phase of the process ontology development focuses on modeling the parameters that facilitate dynamic architectural compositions in a SOA based on the agreed upon process definition. These parameters are modeled as the *architectural weaving rules* that guide the assembly of process-support components with available knowledge services.

As an example, in Fig. 7, the workbench task of "Select Applicable Requirements" (Level 4, Label 2) is associated with an architectural weaving rule "r2" that configures the process-support component "PC_2" to consume the knowledge service "S_1" when the C&A process execution reaches that task. To understand these rules, we first discuss the design of process-support components as consumers of knowledge services in the workbench. Process-support components are essentially rich clients that allow several analytical operations to be performed by utilizing the categorization and classification of compliance evidences retrieved through knowledge services.

5.2.1. *Process-support Component Design*

To promote reuse and reduce coupling of process-support components across available services, we have employed several best design practices and patterns in component design [3]. To consume available services, a process-support component requires contractually specified interfaces as well as agreed upon terminology to share a context of assumptions between them [7]. However, building process-support components limited to the specification of a single service interface prevents the possibility of their reuse. In addition, the component interfaces have to adhere to the granularity of the service interfaces. To address these issues, we define *service connector types* as further classification/categorization of the operations supported by a service interface based on their homogeneity (i.e. relevance to a particular domain concept) or group membership for a particular task. In effect, process-support components programmed to accept *service connector types* can be parameterized with different but conceptually similar services.

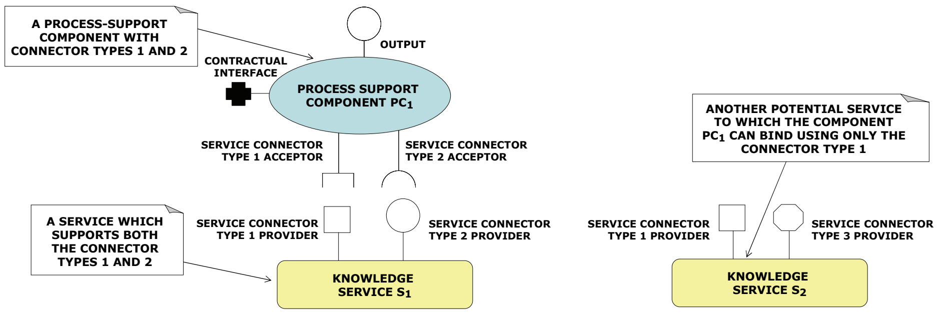
Let us consider an example in the context of the DITSCAP to further understand the use of service connector types. The RDM knowledge service, as discussed in Fig. 3 and Section 4.1, provides operations to access hierarchical *requirements applicability questionnaires* that determine the applicability of regulatory requirements to the system being certified. In addition, the RDM knowledge service provides operations to access non-hierarchical (but categorized) *requirements compliance questionnaires* to gather compliance evidences for the applicable requirements. Therefore, to design a generic questionnaire process-support component (for presenting questions and gathering evidences), we split the operations supported by the RDM knowledge service into two separate "*questionnaire service connector types"*, which handles both types of questionnaires. Essentially, a service connector type acts as an "abstraction" built on top of a single service to provide flexible connectors between the services and the process-support components. The entire service interface itself can be a service connector type (with low possibility of reusing the consuming component) or the generic service connector types can be defined across conceptually similar services (with high possibility of reusing the consuming component). A process-support component is programmed as an *acceptor* of service connector types, whereas services act as a *provider*.

To support dynamic initialization of process-support components at runtime with required services, we have applied the Inversion of Control (IoC) design pattern [3]. The primary rational behind IoC is that, instead of a component requesting to bind with other components/services, the runtime environment calls the component and supplies the resources necessary for it to execute [3]. Therefore, during process-support component design all dependencies to external resources are removed. Then, during architecture configuration at runtime, this information is dynamically supplied to a process-support component. To enable the IoC pattern, all process-support components must support a contractual interface that mandates a certain common expected behavior amongst all participating entities. The contractual interface mandates the following operations to be supported by a process support-component:

- *Interface injection*: The "`setService`" method accepts service name and service connector type as parameters. Through this method, the component is *injected* with the information it requires to be attached with appropriate services that are providers of acceptable service connector types.
- *Associating Task Listener*: A runtime environment subscribes to the process-support component for being notified after task completion. To enable subscription, based on the observer pattern [16], the component (subject) provides the "`addTaskListener`" method. A runtime environment (observer) must implement the "`TaskEventInterface`" interface to be eligible for the subscription.
- *Task Completion Event*: As part of the observer pattern, the process-support component (subject) needs to notify the completion of its task to the runtime

Accepted on July 14, 2008

environment (observer) that subscribes for the notification. The component calls the "`raiseTaskCompletionEvent`" method on itself to perform this notification.



A PROCESS-SUPPORT COMPONENT WITH CONNECTOR TYPES 1 AND 2
OUTPUT
CONTRACTUAL INTERFACE
PROCESS SUPPORT COMPONENT PC$_1$
ANOTHER POTENTIAL SERVICE TO WHICH THE COMPONENT PC$_1$ CAN BIND USING ONLY THE CONNECTOR TYPE 1
SERVICE CONNECTOR TYPE 1 ACCEPTOR
SERVICE CONNECTOR TYPE 2 ACCEPTOR
A SERVICE WHICH SUPPORTS BOTH THE CONNECTOR TYPES 1 AND 2
SERVICE CONNECTOR TYPE 1 PROVIDER
SERVICE CONNECTOR TYPE 2 PROVIDER
SERVICE CONNECTOR TYPE 1 PROVIDER
SERVICE CONNECTOR TYPE 3 PROVIDER
KNOWLEDGE SERVICE S$_1$
KNOWLEDGE SERVICE S$_2$

Fig. 8. A Conceptual Overview of a Process-Support Component and example Knowledge Services to which it can bind using appropriate Knowledge Service Connector Types.

Fig. 8 summarizes our conceptual understanding of a process-support component along with the role of service connector types in determining appropriate/eligible services to which it can bind. In Fig. 8, the component "PC$_1$" can be injected with the information to bind with knowledge service "S$_1$" or "S$_2$" using appropriate connector types. It should be noted that all connector type inputs accepted by a process-support component may not be required for its execution. The input connector types can also be mutually exclusive. Such constraints are documented in the component specification and enforced at runtime to prevent conflicts.

5.2.2. *Architectural Weaving Rules*

To guide runtime architectural composition, each workbench task instance in the process ontology is associated with one or more architectural weaving rules. The rule associates a process-support component with an appropriate knowledge service and supplies this information at runtime for weaving the integrated workbench architecture. Computationally, depth-first navigation of the hierarchical process ontology and the sequence of the workbench tasks, determines the firing sequence of these rules. Depth-first navigation and sequencing of workbench tasks ensures that all pre-requisite process tasks have been satisfied before reaching a certain point in the process execution.

The structure of an architectural weaving rule is analogous to the *aspect* construct in the AspectJ language [36] [18]. Each rule modularizes architecture composition knowledge in the scope of a workbench task. This knowledge is usually related to the initialization of a process-support component by supplying it with references to the services required for its execution. In addition, it can be used to directly invoke specific events in process-support components or call methods defined in the services at appropriate points in the process execution. Fig. 9 shows the architectural weaving rule structure using the UML modeling notation. The descriptions of elements in Fig. 9 are as follows:

Accepted on July 14, 2008

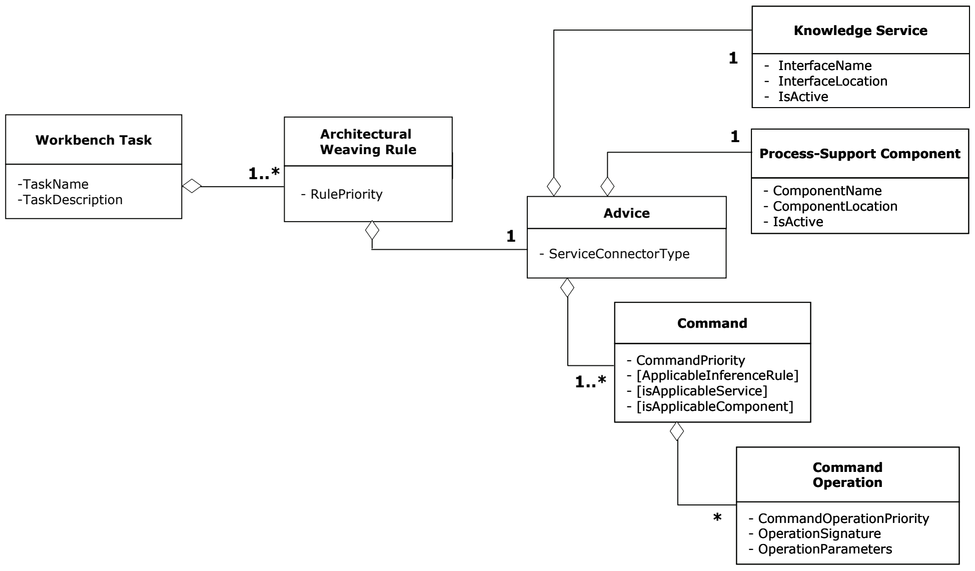


Fig. 9. The Architectural Weaving Rule Model.

- **Architectural Weaving Rule:** A workbench task in the process ontology aggregates one or more architectural weaving rules. A *RulePriority* attribute for each rule determines the execution sequence among multiple rules.
- **Advice:** Each architectural weaving rule aggregates a single *Advice*. An advice maintains references to a single process-support component and a single service. The *ServiceConnecterType* attribute of the advice identifies a compatible match between the referenced process-support component and knowledge service.
- **Command:** An advice aggregates one or more *Commands*. The following properties are associated with a Command:
  - *CommandPriority:* it determines the order of execution among multiple Commands.
  - *ApplicableInferenceRule* (optional): To promote flexibility in using the knowledge base, the *ApplicableInferenceRule* property of a Command can specify an inference to be executed on the knowledge base. The results of the inference rule (inferred tuples) is consumed by the process-support component associated with the Advice.
  - *isApplicableComponent* (optional): A Boolean value, which if true determines that the Command only applies to the Process-support component associated with the Advice.
  - *isApplicableService* (optional): A Boolean value, which if true determines that the Command only applies to the service associated with the advice.

    The *ApplicableInferenceRule*, *isApplicableComponent* and *isApplicableService* properties are mutually exclusive, i.e. only one of them can be valid for a Command.

Accepted on July 14, 2008

- **Command Operation:** A Command aggregates zero or more *Command Operations*. If any one of the *isApplicableComponent* or the *isApplicableService* property of the Command is true, then a *Command Operation* specifies the Operation to be invoked on the process-support component or the knowledge service using the *OperationSignature* and *OperationParameters* attributes. If a Command has an *ApplicableInferenceRule* property then a Command Operation is not required.
  - *CommandOperationPriority*: Prioritizes the execution of Command Operations.
  - *OperationSignature*: This property holds the actual method signature defined in the process-support component interface or the knowledge service interface.
  - *OperationParameters:* This property holds a list of parameters required by the method signature defined in the *OperationSignature* property.

Typically, the *Command Operations* initialize a process-support component and supply it with references to the services that it requires to execute. The *Command Operations* can also be executed directly on a service interface, to perform background tasks. For example, in the context of DITSCAP RDM knowledge service (Fig. 3 and Section 4.1), after the user has answered a *requirements applicability questionnaire*, the next workbench task is to *search all applicable requirements* from the RDM. This task can be initiated by modeling a command which invokes the "selectApplicableRequirements" operation defined in the RDM Knowledge Service "S_1", as show in Fig. 3, Label 1. The specification of *OperationSignature* thus follows the methods defined in the process-component contractual interface or the knowledge service interface. If a process-support component requires multiple service connector types for its execution, then multiple rules are modeled to achieve the required composition. The rules can also be ordered using the *RulePriority* attribute. An example of such composition is demonstrated in Fig. 10.

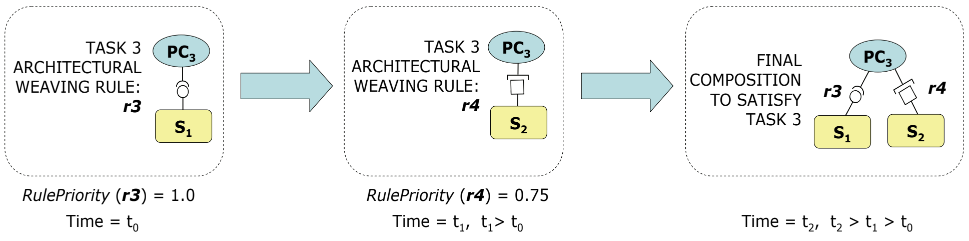


Fig. 10. Multiple Architectural Weaving Rules

6. The Process-Aspect Knowledge Service

The *process-aspect knowledge service* provides operations to access the classification and categorization of the C&A process definition modeled in the process ontology. The process-aspect knowledge service definition allows the process related crosscutting concerns to be injected by a weaving mechanism at specific points during process execution in the workbench architecture. Therefore, we distinguish an *aspect knowledge service* from the knowledge services discussed in Section 4. An aspectual knowledge

Accepted on July 14, 2008

service guides the composition of other services and exposes such knowledge as executable operations for runtime architecture configuration in a particular instantiation.

The process ontology development discussed in the previous section is an ongoing activity as the development of the software system being certified progresses and/or the understanding of the C&A process matures. In tandem, the C&A process execution and the workbench architecture composition continue to perform one workbench task after another until all the high-level goals modeled in the process ontology are satisfied. Although we do not expect the process ontology to be available in its entirety from the start; architecture composition of the workbench can progress using the process-aspect knowledge service as parts of process ontology become available. Pre-engineered templates of process ontology exposed through a process-aspect knowledge service can also address the needs of known variations in the C&A process. With minor adjustments, the pre-engineered templates of the process ontology can readily cater to the C&A needs of different agencies or software system development strategies within an organization.

The process of designing a process-aspect knowledge service interface is similar to that discussed for other knowledge services in Section 4. Independent of the concepts in the process ontology, the process-aspect knowledge service interface supports operations for navigating the process ontology and retrieving an ordered set of workbench tasks to be executed. In the following section we discuss how a runtime environment uses these operations to perform architecture composition.

6.1. *Architecture Composition Algorithm*

To perform a dynamic composition of the workbench architecture, a runtime environment needs to systematically interpret the process ontology exposed through the process-aspect knowledge service. Essentially, we have developed an architecture composition algorithm that provides the runtime environment with an explicit sequence of steps to extract, interpret and fire the architectural weaving rules available from the process-aspect knowledge service. We describe the algorithm using pseudo code in Fig. 11. The first step of the architecture composition algorithm is to identify a sequence of tasks that need to be executed to satisfy the C&A goals. Depth-first navigation of the hierarchical levels defined in the process ontology yields an ordered set of atomic tasks. This type of navigation ensures that pre-requisite tasks are satisfied before reaching a particular task in the process execution. In addition, depth-first navigation does not demand completeness on part of the process ontology. Depth-first navigation is accomplished by the "`getAtomicTasks`" method defined in the process-aspect knowledge service interface (Fig. 11, Line 4).

The next step is to identify the set of workbench tasks modeled for each atomic task (Fig. 11, Line 7). Since most knowledge base operations return an unordered set of results, the retrieved set of workbench tasks are explicitly ordered based on the "predecessor_of" relationship among them (Fig. 11, Line 8) to initialize their execution sequence (Fig. 11, Line 9).

Accepted on July 14, 2008

```
1.  Begin: Architecture Composition Algorithm
    //Initialization
2.  WorkbenchExecutionPool W ← null;
3.  KnowledgeInterface P ←  KnowledgeFactory.getKnowledgeInterface
                                    ("ProcessAspectKnowledgeService")
4.  AtomicTaskList L ←  P.getAtomicTasks();
5.  WorkbenchTaskList T ← null;
    //Workbench Task List Identification
6.  for each AtomicTask A in the AtomicTaskList L, do
7.     WorkbenchTaskList V ← P.getWorkbenchTasksList(A);
8.     V ← V.sortUsingInterdependencies();
9.     T.append(V)
10. end for
    //Dynamic Workbench Composition
11. for each WorkbenchTask t in WorkbenchTaskList T, do
12.    WeavingRuleList R ←  P.getWeavingRuleList(t);
13.    R ← R.sortByRulePriority();
       // Aspect-Rule Interpretation
14.    CompositionKnowledge C ← null;
15.    for each WeavingRule r in WeavingRuleList R, do
16.        C.append(P.interpretWeavingRule(r));
17.    end for
14.    //Perform Architecture Composition and
       //Save Execution Context for Session Management
18.    C.execute();
19.    W.add(C);
       //Assign Task Listener
20.    ProcessSupportComponent X ← C.getComponent();
21.    X.addTaskListener(RuntimeEnvironment);
       //Proceed upon Task Completion
22.    if X.raiseTaskCompletionEvent() Equals True, then
23.        Notify RuntimeEnvironment;
24.        Proceed;
25.    end if
26. end for
27. end algorithm
```

Fig. 11. Workbench Composition Algorithm executed by the Run-time Environment

Several steps are involved in processing each identified workbench task in the execution sequence. The first step is to retrieve and order the set of architectural weaving rules associated with a workbench task (Fig. 11, Lines 12 and 13). Then each rule is interpreted to gather the composition knowledge necessary to weave the workbench architecture (Fig. 11, Line 16). Using this composition knowledge the appropriate process-support

Accepted on July 14, 2008

component and knowledge services are invoked by the runtime environment to setup the necessary conditions required for workbench task execution (Fig. 11, Line 18). This execution is succeeded by the runtime environment setting up a task listener on the invoked process-support component (Fig. 11, Line 21) which then notifies the runtime environment of the completion of its task (Fig. 11, Lines 22 and 23). The runtime environment proceeds with the next workbench task once the task completion notification is received. The algorithm completes when all the workbench task instances have been executed to produce as output an integrated workbench architecture that conforms to the C&A process definition.

6.2. *Stakeholder Understanding of the Process-Aspect Knowledge Service*

The architectural style described in this paper has been enabled in a prototype C&A workbench implementation to support the DITSCAP. The prototype system addresses the DITSCAP automation objectives for understanding C&A requirements applicability, compliance evidence gathering, risk analytics, process analytics, and documentation analytics [47] [46] [38]. Although the scope of this paper does not permit a detailed discussion of these design objectives, here we elaborate upon our efforts to leverage the process ontology to increase stakeholder understanding of the process definition and its implementation using SOA.

Fig. 12 depicts a well-annotated screenshot of the "Process Understanding" interface in the workbench that provides several insights to comprehend the tailored process definition as well as its implementation using SOA. In particular, stakeholders can browse the hierarchical organization of process activities in the process ontology (Fig. 12, Label 2) and list the associated workbench tasks (Fig. 12, Label 3); knowledge services (Fig. 12, Label 4); and process-support components (Fig. 12, Label 5). This explicit traceability helps develop metrics for the effectiveness of the knowledge services as well as their reuse across process activities. These metrics are important to justify the functional and non-functional characteristics of a SOA. Although the current prototype implementation provides simple visualization techniques (Fig. 12, Label 8) to browse the process definition, in the future, we plan to add more support to visualize interdependencies among process activities, process tracking and systematically navigate the artifacts produced throughout the process lifecycle.

The architectural weaving rules (Fig. 12, Label 6 and 7) associated with each workbench task provide in-depth understanding of required compositions in the SOA to satisfy the task. Such rule browsing systematically reveals the required dynamics and flexibility in architecture composition to support a complex and tailorable C&A process.

7. Related Work

In the context of web-services, planning the service selection and interaction are the most significant parts of executing a task defined in an abstract business process workflow [39] [8]. This notion of enabling a process workflow execution through interactions among

Accepted on July 14, 2008

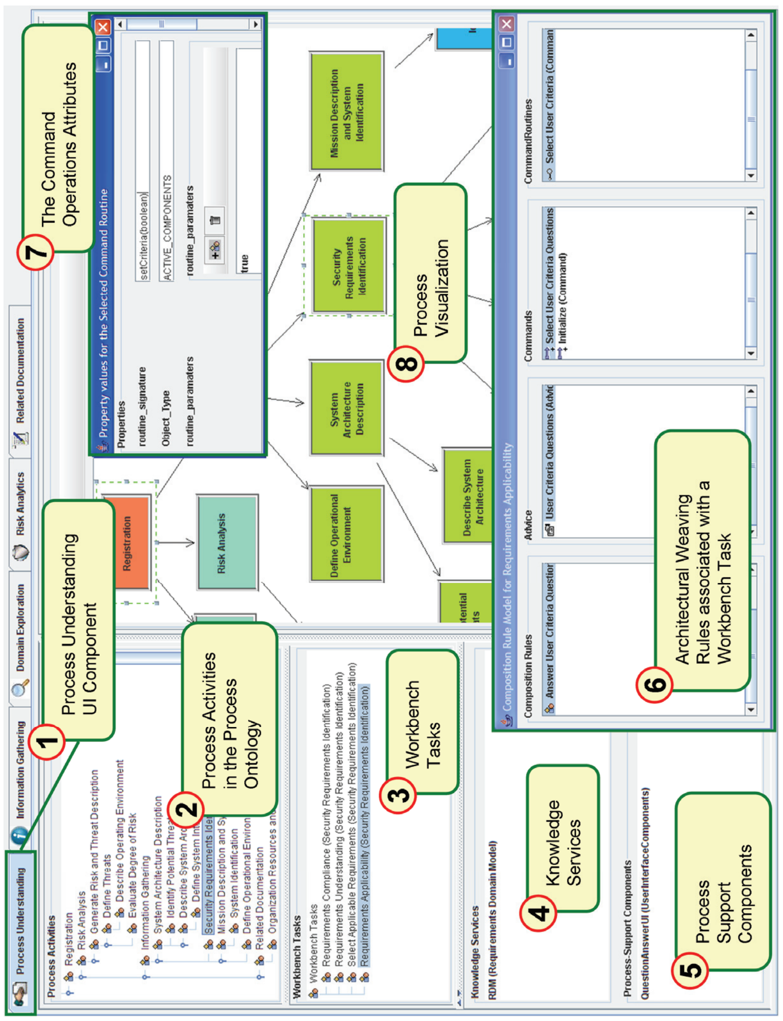

1 Process Understanding UI Component
2 Process Activities in the Process Ontology
3 Workbench Tasks
4 Knowledge Services
5 Process Support Components
6 Architectural Weaving Rules associated with a Workbench Task
7 The Command Operations Attributes
8 Process Visualization
Process Understanding
Information Gathering
Domain Exploration
Risk Analytics
Related Documentation
Registration
Risk Analysis
Define Operational Environment
System Architecture Description
Security Requirements Identification
Mission Description and System Identification
Describe System Architecture
Composition Rule Model for Requirements Applicability
Property values for the Selected Command Routine

Fig. 12. A Screenshot of the Process Understanding Tab in the C&A Workbench Prototype for the DITSCAP

selected services has received much attention in the grid community [20] [55]. Naturally, in order to automate planning of service selection and interaction, the usage of semantic information about services is gaining momentum as the next logical step in the evolution of web services [30] [50] [26] [23]. These solutions address the problems related to web services fueled by the growth of the Internet: 1) how to select the best set of services among numerous available services based on factors such as process constraints, end-user preferences, execution context or other QoS constraints [8]; and 2) how to manage interactions among the selected heterogeneous services. Rather than focusing on the selection and interactions among services available in a web environment, we focus on providing solutions to compose applications using SOA that cater to the need for tailorable processes in a socio-technical environment. While pointing to the view of web

Accepted on July 14, 2008

services as the implementation environment for process enactment, Hollingsworth [10] stresses that such a view raises the danger of ignoring the organizational and human aspects of the business process, in favor of a resource model entirely based on web services. Particularly, complex processes, such as C&A are deep-rooted within a rich organizational and social setting, therefore their execution cannot be completely specified in terms of service interactions but based on a continuous exchange of artifacts between the social and technical worlds separated at the process-support system boundary.

Benefits of embedding descriptive ontologies or referencing semantic metadata within software system design models are being advocated in a recent World Wide Web Consortium (W3C) working draft [34]. The rationale is to combine semi-formal, model-driven techniques of software engineering with approaches common to knowledge engineering. These principles have also been applied to support the development and administration of software components deployed in an application server using formal ontological definitions [12]. In this approach, the ontology construction is geared towards facilitating system administrators or developers in application server configuration by providing concepts that describe component and service characteristics. However, this approach lacks a meta-model that links the specific components and services with the high-level strategic business process goals that justify their existence and appropriate composition. Such high-level goals also have been deemed important in the frameworks proposed for self-managed and dynamic adaptive systems [28]. The applications of these systems in robotics [11] have shown the use of component-based software development and ontologies to enable dynamic architecture reconfiguration in response to changes in environmental factors. Cervantes et al. [22] have introduced concepts from service orientation into a component model to build autonomic component-based applications that react to changes in service availability. However, their framework does not focus on issues related to stakeholder understanding of the composed application architectures or separation of process concerns.

The synergy between aspect and service oriented paradigms has also been explored by Mendonca et al. [32]. They introduce the notion of *aspectual services* which invoke additional behavior upon identifying a particular message interaction between the service consumer and service provider. However, these aspectual services are loosely coupled and do not have any impact on the architecture of the system. In our approach, the process ontology allows for early requirements engineering artifacts (e.g. early aspects [4] in requirements engineering) to be carried over to later service lifecycle stages as well as influence service deployment. The process ontology supports explicit traceability between problem-level abstractions and their implementations using services, a notion which is central to the philosophy of Model-driven Engineering (MDE) [35].

Our work in many aspects complements the principles of the Workflow Management Coalition (WfMC) for supporting business process workflows [9]. The BPEL language in the webservices domain has also originated from the general principles of WfMC. More recent extensions of BPEL [25] such as AO4BPEL [2] and Aspect Weaving BPEL Engine [5] focus on modularization of the BPEL specification to support dynamic

Accepted on July 14, 2008

adaptation capabilities of the previously static process definition. In contrast, our approach addresses the problems of process complexity and tailorability based on rich and flexible knowledge models of the process definition that are built to reflect customized needs and are later exposed as services themselves. Our approach promotes homogeneity in the service-oriented infrastructure by eliminating the need for specialized engines to interpret and manage process specifications.

Knowledge based systems have long been advocated to elicit, represent and disseminate rich information throughout the software lifecycle [6], for example in modeling and analyzing requirements [21] [29] [43] [6], capturing design rationale [24], and many others. However, no systematic guidance is available in designing flexible and scalable applications. Our work provides important insights for exposing ontological models that offer deep representation of the software artifacts themselves and then exposing such knowledge as services that support the enactment of a process.

8. Contributions and Future Work

In this paper we discussed the challenges in composing applications based on SOA to support business processes that are complex (highly embedded in a socio-technical environment) and tailored (based on stakeholder negotiations) in each of their instantiations. We demonstrated by example of a C&A process, our approach to systematically capture the process definition as an ontological model and expose it as a process-aspect knowledge service that guides architecture composition based on the principles of SOA. While enumerating the steps in our approach, we identify several best practices in service and component design for a dynamic and flexible SOA. As an intended benefit of our approach we demonstrate strategies for end-users to understand and possibly configure SOA based process-support systems using intuitive interfaces and visualizations.

Our approach for composing applications using SOA has contributed to several desirable characteristics in the design of a workbench for supporting C&A processes. The C&A process is a huge undertaking which requires enormous amount of resources to define, conduct and manage. To this end, the workbench architecture provides flexibility in configuring a tool support that is tailored to meet the needs of the C&A process in each of its instantiations. From the perspective of the C&A process, the work presented here makes the following contributions:

- The workbench architecture is composed in accordance with the on-going tailoring effort of the C&A process
- The process ontology provides active guidance to stakeholders through explicit traceability between the C&A process definition and the available services and components
- Early separation of C&A process related cross-cutting concerns is achieved using ontological domain modeling techniques. This separation allows the process, services and components in the architecture to be loosely coupled with each other.

Accepted on July 14, 2008

- The process ontology as a hierarchical decomposition of high level strategic process goals maintains an explicit argument for process fulfillment through automation using SOA and allows fine-grained progress tracking
- The process ontology construction helps promote a common understanding among stakeholders of a complex and exhaustive C&A process. The inherent richness and flexibility offered by ontological domain modeling techniques imposes little or no constraints while representing the granularity or scope of the original C&A process definition
- We identify guidelines for exposing ontological domain models as services through a incremental and layered approach to produce interface definitions
- We define architectural weaving rules and demonstrate the use of aspect-oriented design philosophies to weave process related cross-cutting concerns with services that provide functional abstractions
- The architectural weaving rules provide declarative specifications to compose SOA based on semantics that are easily understandable and reviewable by stakeholders
- We outline the design of a unique and integrated solution that leverages the synergy among service, aspect, component, and ontological domain modeling philosophies to build a highly dynamic and flexible SOA

It should be noted that flexibility of our SOA-based design solution depends on several factors which include the richness of the ontological models in the PDO, the granularity of tasks supported by process-support components, and the level of abstraction of the operations supported by the knowledge services. Nevertheless, our approach provides the necessary techniques to address the needs of a complex and tailorable process through dynamic and transparent (end-user participation and understanding) configuration of its process-support system architecture.

As part of our ongoing and future work, we are exploring the opportunities offered by SOA to support C&A in a net-centric environment [19]. A net-centric environment requires faster assess to current C&A information, at a reduced cost, and delivered simultaneously to a variety of devices in different locations. The vision of net-centric C&A is currently seen as "*networked C&A activities accomplished through distributed collaboration processes designed to ensure that all pertinent available system-security information is dynamically managed, visible, and shared*" [19]. To facilitate this vision, existence of a common understanding of regulatory requirements among all the distributed collaborating C&A and risk assessment processes is inevitable. In addition, tool support for C&A should be able to aggregate and deliver artifacts from heterogeneous sources in a distributed environment. To this end, the work presented in this paper provides a guidance to develop services to aggregate, produce, analyze and disseminate C&A artifacts using ontological domain modeling techniques. The recent transition of DITSCAP to DIACAP [19], geared towards net-centric infrastructure, implies the changes in process and the format of delivery and consumption of C&A artifacts; but they still significantly overlap over the set of documents suggested for identifying C&A requirements. This case also confirms the stability and reusability of the knowledge services defined for a C&A process in an organization.

Accepted on July 14, 2008

Our prototype workbench provides an excellent test-bed for further exploring the possibilities to separate other functional and non-functional cross-cutting concerns related to access control, dynamic help, version control, logging, and accountability. These are all important concerns while composing an application based on SOA. We are also working towards managing access to services or parts of services (groups of critical operations) based on user roles and access control policies. Providing runtime evaluation of the interactions among multiple cross-cutting concerns to prevent conflicting architectural compositions is also another important direction for our future work. Finally, we plan to perform a case study using domain experts to evaluate the impact our design on process understanding and the ease of architecture configuration while tailoring the C&A process.

9. References

[1] A. Brown, S. Johnston and K. Kelly, Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications, *Rational Software White Paper*, (2002)

[2] A. Charfi and M. Mezini, "Aspect-Oriented Web Service Composition with AO4BPEL," In Proc. of European Conference on Web Services, LNCS Vol. 3250, Springer-Verlag, 2004

[3] A. Dearle, Software Deployment: Past, Present and Future, In *Proc. of the Future of Software Engineering, held in conjunction with the 29th Int. Conf. on Software Engineering*, (Minneapolis, MN, 2007), pp. 269-284.

[4] A. Rashid, P. Sawyer, A. Moreira and J. Araujo, Early aspects: a model for aspect-oriented requirements engineering, In *Proc. of IEEE Joint Int. Conf. on Requirements Engineering*, (9-13 Sept. 2002), pp. 199 – 202.

[5] C. Courbis, and A. Finkelstein, "Towards an Aspect Weaving BPEL Engine," In *Proc. of the 3rd AOSD Workshop on Aspects, Components, & Patterns for Infrastructure Software*, (2004)

[6] C. Rich, and R. C. Waters, Knowledge Intensive Software Engineering Tools, *IEEE Transactions on Knowledge and Data Engineering*, **4**(5), (1992) 424-430.

[7] C. Szyperski, Component technology: what, where, and how?, In *Proc. of the 25th Int. Conf. on Software Engineering*, (Portland, Oregon, 2003), pp. 684 - 693.

[8] D. Ardagna and B. Pernici, Adaptive Service Composition in Flexible Processes, *IEEE Transactions on Software Engineering*, **33**(6) (2007).

[9] D. Hollingsworth, The Workflow Reference Model, *WFMC-TC-1003, Ver. 1.1*, (19-Jan 1995), http://www.wfmc.org/

[10] D. Hollingsworth, The Workflow Reference Model: 10 Years On, *Book Chapter in the Workflow Handbook*, (2004), http://www.wfmc.org/standards/referencemodel.htm

[11] D. Kim, S. Park, Y. Jin, H. Chang, Y. Park, I. Ko, K. Lee, J. Lee, Y. Park and S. Lee, SHAGE: a framework for self-managed robot software, In *Proc. of the Int. workshop on Self-adaptation and self-managing systems at the Int. Conf. on Soft. Engg.*, (2006), pp. 79 – 85.

[12] D. Oberle, S. Staab and A. Eberhart, Towards Semantic Middleware for Web Application Development, *IEEE Distributed Systems Online* (2005).

[13] DoD 8510.1-M, DITSCAP Application Manual, (2000).

[14] DoD Instruction 5200.40: DITSCAP, (1997).

[15] DoD Instruction 8500.2. IA Implementation. (Feb 2003).

[16] E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, (Addison -Wesley, 1995).

Accepted on July 14, 2008

[17] G. Alonso, F. Casati, H. Kuno and V. Machiraju, *Web Services. Concepts, Architectures and Applications*, (Springer-Verlag, Berlin Heidelberg, 2004).

[18] G. Kiczales and M. Mezini, Aspect-oriented programming and modular reasoning, In *Proc. of the 27th Int. Conf. on Software Engineering, (ICSE '05),* (2005), pp. 49- 58.

[19] G. Turner, P. Holley, E.J. Mehan and M. Colon, Net-Centric Assured Information Sharing – Moving Security to the Edge through dynamic C&A, *IANewsletter*, **8**(3), (Winter 2005/2006).

[20] G.C. Fox and D. Gannon, Workflow in Grid Systems, *Int. J. on Concurrency and Computation: Practice and Experience*, **18**(10), (Wiley, 2006) 1009-1019.

[21] H. B. Reubenstein and R. C. Waters, The Requirements Apprentice: Automated Assistance for Requirements Acquisition, *IEEE Trans. on Soft. Engg.*, **17**(3), (Mar. 1991) 226-240.

[22] H. Cervantes and R.S. Hall, Autonomous adaptation to dynamic availability using a service-oriented component model, In *Proc. of the 26th Int. Conf. on Soft. Engg.,* (2004), pp. 614- 623.

[23] I. Budak Arpinar, Alman-Meza, R. Zhang and A. Maduko, Ontology driven Web Service Composition Platform, *IEEE Int. Conf. on E-Commerce Technology*, (2004).

[24] I. Gorton and A. Babar, Architecture Knowledge Management: Concepts, Technologies, Challenges, *The Working IEEE/IFIP Conf. on Software Architecture*, (2007).

[25] IBM, BEA Systems, Microsoft, SAP AG and Siebel Systems, Business process execution language for web services (BPEL4WS) version 1.1. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/, (2005).

[26] J. Cardoso and A. Sheth, Semantic E-Workflow Composition, *J. of Intelligent Information Sys.* **21**(3) (2003) 191-225

[27] J. Kimbell and M. Walrath, Life Cycle Security and DITSCAP, *IANewsletter*, **4**(2), (2001) http://iac.dtic.mil/iatac

[28] J. Kramer and J. Magee, Self-Managed Systems: An Architectural Challenge, In *Proc. of the Future of Software Engineering, held in conjunction with the 29th Int. Conf. on Software Engineering*, (Minneapolis, MN, 2007), pp. 259-268.

[29] J.C.S.P. Leite and P. A. Freeman, Requirements Validation through Viewpoint Resolution, *IEEE Transactions on Software Engineering*, **17**(12), (Dec. 1991) 1253-1269.

[30] M. Meyer and D. Kuropka, Requirements for Service Composition, *Tech. Report of the Hasso-Plattner-Institute, Partners of the Automated Services Grid (ASG) Project 11* (2005), ISBN 3-937786-81-3

[31] M. P. Papazoglou, and D. Georgakopoulos, Service-oriented computing, *Communications of the ACM (CACM)* **46**(10), (Oct. 2003).

[32] N.C. Mendonca and C.F. Silva, Aspectual Services: Unifying Service-and Aspect-Oriented Software Development, In *Proc. of Int. Conf. on Next Gen. Web Services Practices*, (2005).

[33] Office of Management and Budget (OMB), FY 2005 Annual Report to Congress on Implementation of The Federal Information Security Management Act of 2002, (March 1, 2006).

[34] P. Tetlow, J.Z. Pan, D. Oberle, E. Wallace, M. Uschold and E. Kendall (eds.), Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering, *Draft*, (Feb 2006), http://www.w3.org/2001/sw/BestPractices/SE/ODA/

[35] R. France and B. Rumpe, Model-Driven Development of Complex Software: A Research Roadmap, In *Proc. of the Future of Software Engineering, held in conjunction with the 29th Int. Conf. on Software Engineering*, (Minneapolis, MN, 2007), pp. 37-54.

[36] R. Laddad, *AspectJ in action: practical aspect-oriented programming,* (Manning Publications, 2003).

[37] R. Perrey and M. Lycett, Service-oriented architecture, In *Proc. of Symp. on Applications and the Internet Workshops (SAINT '03)*, (2003), pp. 116- 119.

Accepted on July 14, 2008

[38] R.A. Gandhi, and S.W. Lee, Discovering and Understanding Multi-dimensional Correlations among Certification Requirements with application to Risk Assessment, *In Proc. of the 15th IEEE Int. Requirements Engg. Conf.*, (October 15-19, Delhi, India, 2007) pp. 231-240.

[39] S. Dustdar and W. Schreiner, A survey of web services composition, Int. *J. of Web and Grid Services*, **1**(1), (2005) 1-30.

[40] S. K. Miller, Aspect-oriented programming takes aim at software complexity, *IEEE Computer*, **34**(4) (Apr 2001) 18-21.

[41] S.W. Lee and D. Yavagal, GenOM User's Guide V2.0, *Tech. Report TR-NiSE-05-05*, (UNC Charlotte, 2005).

[42] S.W. Lee and R.A. Gandhi, Requirements as Enablers for Software Assurance, *CrossTalk: The Journal of Defense Software Engineering*, **19** (12, December issue), (2006) 20-24.

[43] S.W. Lee, and R.A. Gandhi, Ontology-based Active Requirements Engineering Framework, In *Proc. 12th Asia-Pacific Soft. Engg Conf.*, (IEEE CS, 2005), pp. 481-490.

[44] S.W. Lee, D. Muthurajan, R.A. Gandhi, D.S. Yavagal and G.J. Ahn, Building decision support problem domain ontology from natural language requirements for software assurance, *Int. J. on Software Engg and Knowledge Engg.*, **16**(6), (2006) 851-884.

[45] S.W. Lee, R.A. Gandhi and G.J. Ahn, Certification Process Artifacts Defined as Measurable Units for Software Assurance, *Int. J. on Software Process: Improvement and Practice*, *John Wiley & Sons, Ltd.* **12**(2), (2007) 165-189, doi: http://dx.doi.org/10.1002/spip.313

[46] S.W. Lee, R.A. Gandhi, S.J. Wagle, and A.B. Murty, r-AnalytiCA: Requirements Analytics for Certification & Accreditation, *In the Proc. of the 15th IEEE Int. Requirements Engg. Conf. (RE 07), Posters, Demos and Exhibits Session*, (October 15-19, Delhi, India, 2007).

[47] S.W. Lee, R.A. Gandhi, and S.J. Wagle, Towards a Requirements-driven Workbench for Supporting Software Certification and Accreditation, In *Proc. of the 3rd Int. Workshop on Software Engineering for Secure System (SESS 07), at the 29th Int. Conf. on Software Engineering*, (Minneapolis, MN, 2007).

[48] S.W. Lee, R.A. Gandhi, and G.J. Ahn, Security Requirements Driven Risk Assessment for Critical Infrastructure Information Systems, *In Proc. Symp. on Requirements Engg. for Information Security (SREIS '05), at 13th IEEE Int. Requirements Engg. Conf. (RE '05)*, (Paris, France, IEEE Press, 2005).

[49] S.W. Lee, R.A. Gandhi, and S. Park, *Tracing Requirements*, (a book chapter to appear in The Encyclopedia of Software Engineering, Taylor and Francis Group, LLC, 2008)

[50] T. Fahringer, H. Krause, D. Kuropka, H. Mayer,A. Ocampo, B. Schreder, S. Staab, P. Tröger, A. Wahler and M. Zaremba, ASG technology advantages and disadvantages, exploitation possibilities and its business impact, *Adaptive Services Grid – White Paper*, (March 19th, 2007), http://asg-platform.org/cgi-bin/twiki/view/Public/WebHome

[51] The United States General Accounting Office, Agencies Need to Implement Consistent Processes in Authorizing Systems for Operation, *Report to Congressional Requestors, GAO-04-376,* (June 2004)

[52] V. K. Chaudhri, A. Farquhar, R. Fikes, P.D. Karp and J. Rice, OKBC: a programmatic foundation for knowledge base interoperability, In *Proc. 15th Conf. on Artificial Intelligence*, (1998), pp. 600-607.

[53] van Lamsweerde, A., Goal-oriented requirements engineering: a guided tour, In *Proc. of the 5th Int. Symp. on Requirements Engg.*, (Toronto, Canada, August 2001), pp. 249-262.

[54] W. Swartout; A. Tate, "Ontologies" *IEEE Intelligent Systems*, **14**(1), (1999) 18-19.

[55] Z. Laliwala, R. Khosla, P. Majumdar and S. Chaudhary, Semantic and Rules Based Event-Driven Dynamic Web Service Composition for Automation of Business Process, *In Proc. of IEEE Services Computing Workshop,* (2006).