Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Administración de flujos de trabajo organizados en áreas autónomas para entornos ubicuos

Tesis que presenta

Madai Navarrete Castro

para obtener el Grado de

Maestro en Ciencias

en Computación

Director de la Tesis

Dra. Sonia Guadalupe Mendoza Chapa

México, D.F. **Febrero 2010**

Resumen

En la actualidad, las grandes organizaciones están divididas en sub-organizaciones (e.g., departamentos, servicios administrativos y secciones) las cuales incorporan tecnologías de la información para la gestión de algunos de los distintos procesos que se llevan a cabo dentro de ellas. Una de las tecnologías más utilizadas se refiere a los sistemas de flujo de trabajo, cuyo objetivo principal es guiar y controlar de forma automática los componentes de un proceso de negocio, e.g., tareas, documentos, normas y aplicaciones. La incorporación de este tipo de sistemas en una organización permite una mejor orquestación de sus recursos. Sin embargo, estos sistemas no han considerado la automatización del enrutamiento de las actividades de un usuario nómada, quien requiere moverse a lo largo de la organización e interactuar con los servicios ofrecidos por sus diferentes áreas, con el fin de lograr sus objetivos. Esta tesis de maestría pretende proveer un soporte de administración de flujos de trabajo en un entorno ubicuo. Los objetivos principales de este soporte son: 1) proveer a los usuarios nómadas de información sobre las actividades que deben llevar a cabo dentro de cada área visitada, y 2) otorgar permisos e imponer restricciones a dichos usuarios con base en las políticas de seguridad de la organización. De esta manera, el sistema propuesto se caracteriza por: a) ofrecer una administración basada en áreas que organiza, de manera jerárquica, las diferentes áreas de la organización de manera que las áreas de niveles superiores provean permisos y restricciones generales, mientras que las áreas de niveles inferiores ofrezcan permisos y restricciones específicas; b) determinar el contexto del usuario nómada dentro del área actual y de otras áreas que puedan influenciar el flujo de trabajo del usuario; c) regular las actividades del usuario nómada; y d) controlar el acceso a los servicios y recursos proveídos por cada área. El flujo de trabajo proporcionado al usuario nómada depende de su actual ubicación, rol y objetivo. Particularmente, la ubicación corresponde al área actual donde se encuentra el usuario nómada; el rol define los permisos que serán otorgados dicho usuario por el área actual; y el objetivo determina las actividades que el usuario debe llevar a cabo dentro de las áreas involucradas. El sistema propuesto pretende ofrecer una administración autónoma de las áreas que componen la organización, de tal manera que cada una de ellas se encargue de la administración de sus recursos, información y servicios, así como de generar el flujo de trabajo del usuario nómada en función de su contexto actual.

Palabras clave: cómputo ubicuo, flujos de trabajo, áreas autónomas, Control de Acceso Basado en Roles.

Abstract

Nowadays, large organizations are divided into sub-organizations (e.g., departments, management services and sections), which incorporate information technologies for managing some of the different processes that are carried out within them. One of the most employed technologies refers to the workflow systems, whose main goal is to guide and control in an automatic way the components of a business process, e.g., tasks, documents, rules and applications. The incorporation of this kind of systems into an organization allows a better orquestation of their resources. However, these systems have not considered the routing automation of the activities of a nomadic user, who requires to move along the organization and to interact with the services provided by its different areas, in order to achieve his goals. This master thesis aims at providing a management support for workflows within a ubiquitous environment. The main goals of this support are: 1) to provide nomadic users with information about the activities they have to carry out within each visited area, and 2) to grant rights and to impose restrictions on such users based on the security policies of the organization. In this way, the proposed system is characterized by: a) offering an area-based administration, which hierarchically organizes the different areas of the organization, so that high level areas provide general rights and restrictions, while low level areas offers specific ones; b) determining the nomadic user's context within the current area and other areas that can influence the user's workflow; c) regulating the activities of the nomadic user; and d) controlling the access to the services and resources supplied by each area. The workflow provided to the nomadic user depends on his current location, role and goal. Particularly, the location corresponds to the current area where the nomadic user is situated; the role defines the rights granted to this user by the current area; and the goal determines the activities the user must carry out within the involved areas. The proposed system aims at providing an autonomous administration of the organization areas, so that each one is responsible for managing its own resources, information, and services, and generating the nomadic user's workflow based on his current context.

Keywords: ubiquitous computing, workflows, autonomous areas, Role-Based Access Control

Agradecimientos

A Dios, porque ha sido, es y será mi guía. Por haber puesto en mi camino a todas aquellas personas que han sido mi soporte y compañía durante esta etapa de estudio y sobre todo por haberme permitido culminar satisfactoriamente mis metas académicas.

A mis asesores, mi más profundo agradecimiento y admiración...

Dra. Sonia Mendoza Chapa:
Gracias por haberme dado la oportunidad y brindado la confianza para realizar este trabajo de tesis. Por el tiempo, esfuerzo y dedicación que ha implicado dirigir este trabajo y sobre todo por transmitirme sus conocimientos y experiencias que me han permitido crecer no solo en lo académico sino como persona. Muchas gracias.

A mis revisores:
Dr. José Guadalupe Rodriguez y Dr. Dominique Decouchant por el tiempo dedicado a la revisión de esta tesis, sobre todo por compartir sus conocimientos y por motivarme a seguirme superando

Agradezco al Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional por brindarme la formación y las herramientas suficientes durante mis estudios de maestría.

Gracias al Consejo Nacional de Ciencia y Tecnología por haberme proporcionado el apoyo económico para sustentar mi estancia durante el tiempo que realicé mis estudios de maestría.

Sofía Reza:
Gracias por el apoyo y cariño que me brindó durante todo este tiempo, y sobre todo gracias por ver en cada uno de los estudiantes un hijo más.

A mis compañeros:
Agradezco a mis compañeros Pau, Johny, Pam, July, Ray, Lil, Cheche, Miguelin, Lupita, Gox, Beto y Desi, por esos ratos de alegrias que compartimos

Gracias a mi familia por estar siempre apoyandome y motivandome...

A mis papas: *Por ser la parte fundamental de mi vida, por ser mi más grande ejemplo de lucha y trabajo, por sus enseñanzas y consejos que han sido de gran ayuda para mi vida y crecimiento. Por ello esta tesis va dedicada a ustedes, nunca olviden que mis logros son sus logros. Gracias por todo ¡Los amo!.*

A mis hermanos:
Gracias por su cariño, sus consejos, su apoyo, por preocuparse de su hermanita, por compartir sus experiencias de la vida, pero sobre todo, gracias por estar en otro momento tan importante de mi vida.

A Christian:
Gracias por ser parte de mi vida y compartir conmigo estos dos anõs, por estar conmigo en tristezas, alegrias, enfermedad y sobre todo por estar ante todo siempre de pie para tomarme de la mano. Gracias por permitirme compartir otro logro más en nuestras vidas. ¡Te amo!.

Índice general

Índice de figuras

Lista de Tablas

Capítulo 1

Introducción

1.1 Contexto de investigación

Este trabajo de investigación centra su interés en el ámbito de una organización, la cual está dividida lógicamente en áreas para ofrecer, de una manera eficiente, servicios especializados a los usuarios nómadas que se desenvuelven en ellas. Cada área es autónoma e independiente, ya que administra sus propios recursos y servicios, intercambia información con otras áreas para coordinarse entre ellas y toma decisiones en función del contexto actual del usuario nómada. Particularmente, este trabajo de maestría se inscribe en los siguientes tres campos de investigación (ver Figura 1.1): el Cómputo Ubicuo (cf. sección 1.1.1), el Control de Acceso Basado en Roles (cf. sección 1.1.2) y los Flujos de Trabajo (cf. sección 1.1.3).

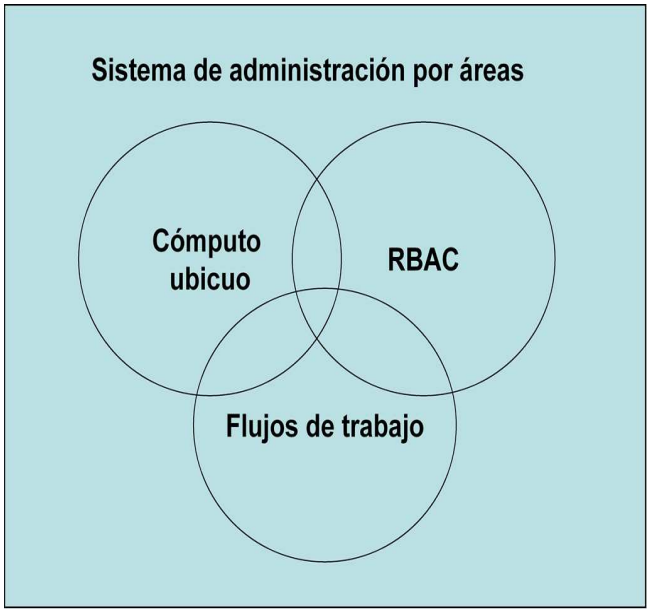

Figura 1.1: Contexto de investigación de la presente tesis

1.1.1 Cómputo ubicuo

Hoy en día, la idea de la computadora personal está quedando poco a poco fuera de lugar debido a que los dispositivos de cómputo emergentes tales como laptops, PDA y *smartphones*, solo son un paso transitorio para lograr un aprovechamiento real del potencial intrínseco de las tecnologías de la información [A., 2005]. Se pretende que tales dispositivos realicen un procesamiento integral y transparente a los usuarios, tomando en cuenta el entorno físico de los usuarios, en primer plano, y la capacidad de procesamiento de las computadoras, en segundo plano.

El gran ideal de Mark Weiser, considerado como el padre del cómputo ubicuo, era que la computadora se convirtiera en un artefacto tan incorporado, adaptable y natural, que se utilizara sin siquiera pensar en ella. La idea principal de Weiser consiste en la desaparición física de la computadora y en la incorporación de capacidades de conmutación y comunicación en la mayoría de los objetos físicos cotidianos, con el fin de formar una gran red de dispositivos interconectados. De esta manera, surge el tercer paradigma de las ciencias de la computación, el ya mencionado cómputo ubicuo, cuyo objetivo es crear entornos físicos inteligentes mediante el uso de sistemas computacionales, que estén disponibles pero que se vuelvan invisibles a los usuarios.

Este nuevo paradigma está dando origen al diseño y a la implementación de sistemas inteligentes que se adaptan al usuario y que se utilizan de manera intuitiva [Weiser, 1993]. Un entorno ubicuo consta de cientos de computadoras colocadas en todas partes dentro de un espacio físico, todas ellas unidas por redes de alta velocidad y capaces de detectar personas, objetos y dispositivos. Este tipo de tecnología requiere de la actualización constante de la información del entorno físico, debido a la movilidad de los usuarios. La aportación más importante del cómputo ubicuo consiste en ayudar a los usuarios a superar el problema de la sobrecarga de información, ya que esta es clasificada con base en el contexto del usuario, antes de ser proporcionada [Weiser, 1999]. La tecnología requerida para implementar el cómputo ubicuo se enfoca en tres puntos: a) computadoras baratas de bajo poder de procesamiento, b) software para aplicaciones del cómputo ubicuo y c) redes para unir los componentes mencionados en los incisos anteriores.

Esta tecnología permite adaptar objetos de distintos tamaños, convirtiéndolos en sensores y actuadores con capacidades de procesamiento, i.e., los dispositivos pueden contener sensores que se encargan de recoger información del entorno y actuadores que modifican dicho entorno y que presentan información al usuario [Aizawa et al., 2004].

Los factores importantes de un entorno ubicuo son la ubicación y la escala, ya que facilitan la percepción humana [Becker and Durr, 2005], e.g., si una computadora sabe lo que hay en una habitación puede adaptar su comportamiento de manera significativa, sin requerir incluso de la inteligencia artificial. Al tener cientos de computadoras interconectadas en un entorno físico y volverse invisibles para la conciencia común del humano, las personas o usuarios pueden utilizarlas inconciententemente para realizar sus tareas [Abowd and Mynatt, 2000].

1.1.2 RBAC

El modelo RBAC[1] es considerado como el modelo de base para la construcción de sistemas de control de acceso en la comunidad de ingeniería de software [Ferraiolo et al., 2001]. RBAC define la estructura de autorización en las organizaciones reales y facilita las tareas administrativas por medio de la asignación de permisos a los usuarios en función de sus roles [Omicini et al.,]. Así, cuando un usuario inicia o entra a una sesión de trabajo se le otorga un determinado rol, el cual activa un subconjunto de permisos sobre los recursos que proporciona la organización.

El modelo RBAC ha sido clasificado en varios submodelos, que están enfocados a organizaciones o entornos distribuidos. Algunos de ellos se describen brevemente a continuación:

- Or-RBAC[2] (Control de Acceso Basado en Roles - Organización) propone una forma segura de administrar una comunidad virtual, dividiendo la organización en suborganizaciones que son administradas independientemente [Cuppens and Miége, 2003];
- RBAC-ADS[3] (Control de Acceso Basado en Roles - Administración en Sistemas Distribuidos) ofrece un sistema de administración centralizada y varios subsistemas que protegen los diferentes subconjuntos de datos, los cuales solo se actualizan si hay cambios relevantes en las políticas [Dekker et al., 2008]; y
- dRBAC[4] (Control de Acceso Basado en Roles - Distribuidos) propone un control de acceso descentralizado, englobando múltiples dominios administrativos que pueden ser utilizados para distribuir, localizar, validar y cancelar gestiones basadas en roles dentro de un contexto de seguridad [Freudenthal et al., 2002].

El modelo RBAC ha sido adaptado a entornos distribuidos, sin embargo los modelos RBAC - ADS y dRBAC no abordan la autonomía del esquema de administración basada en áreas, esquema que concierne nuestro tema de investigación.

La tecnología de flujos de trabajo ofrece la automatización de las tareas de oficina (i.e., sistemas de escritorio) y permite que algunas de estas tareas se realicen de manera colaborativa. La automatización se refiere a la activación de la actividad que el usuario o trabajador debe llevar a cabo, después de que la actividad de otro trabajador haya terminado. De esta manera, el proceso de la tarea a cumplir se realiza satisfactoriamente.

En el dominio de los sistemas de flujos de trabajo, RBAC propone el modelo W-RBAC[5] (Control de Acceso Basado en Roles - Flujos de Trabajo) [Oh and Park, 2002], el cual permite la asignación de roles y tareas a los usuarios, de manera que los permisos son asignados en base a las tareas y al rol del usuario. W-RBAC presenta dos submodelos de RBAC para flujos de trabajo:

- W0-RBAC divide el sistema en distintas funciones que se superponen para reducir la complejidad y facilitar la gestión de un problema; y

[1] *Role - Based Access Control*
[2] *Organizational Based Access Control*
[3] *Administration in Distributed System - RBAC*
[4] *Distributed Role-Based Access Control for Dynamic Coalition Enviroments*
[5] *Workflows - RBAC*

- W1-RBAC extiende el modelo básico para incorporar excepciones en las capacidades de administración de flujos de trabajo.

1.1.3 Flujos de trabajo

Los sistemas de flujos de trabajo (*workflows*) están enfocados en la definición y construcción de procesos de las organizaciones. Estos sistemas determinan los procesos de las organizaciones en término de tareas [Zur Muehlen, 2004]. Los usuarios son definidos como actores, los cuales llevan a cabo un conjunto de tareas siguiendo un orden parcial que define el momento en que deben ser realizadas.

De una manera más formal, un proceso está definido por el flujo de trabajo W, el cual está representado por la tupla (T, P$\prec$), donde T es un conjunto de tareas y P$\prec$ es el orden de T. Por ejemplo, suponga un proceso para llevar a cabo un "reembolso de gastos". Este proceso está definido por la tupla siguiente:

W = (Solicitud, Auditoría, Aprobación 1, Aprobación 2, Reembolso, Rechazo, P$\prec$)

El orden de las tareas se ilustra en la Figura 1.2.

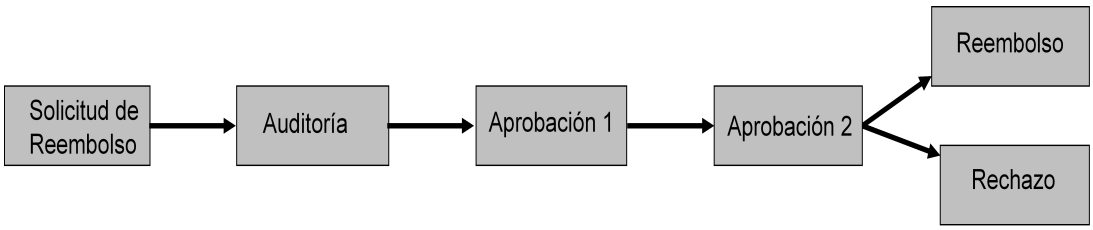

Figura 1.2: Secuencia de pasos para el proceso "reembolso de gastos"

Un proceso de "reembolso de gastos" inicia por la tarea de `Solicitud de Reembolso`, la cual es entonces analizada por la tarea `Auditoría` y finalmente por las tareas de `Aprobación 1 y 2`. Dependiendo de los resultados de las tareas de `Aprobación` se determina si se lleva a cabo la tarea de `Reembolso` o `Rechazo`. El orden de las tareas establece sus restricciones de ejecución, e.g., las tareas `Solicitar`, `Auditoría` y ambas `Aprobaciones` deben ser ejecutadas de manera secuencial, en tanto que las tareas de `Reembolso` y `Rechazo` de reembolso (mutuamente excluyentes) pueden ser ejecutadas inmediatamente después de obtener el resultado de la segunda aprobación.

Normalmente, los flujos de trabajo son una combinación de tareas automáticas y tareas que necesitan de la participación de los humanos. Por ejemplo, en el proceso de reembolso, la mayoría de tareas requieren intervención humana; solo las tareas de `Reembolso` y `Rechazo` son automáticas, las cuales generan un correo electrónico para enviar el resultado del proceso al solicitante. Una vez que se define el flujo de trabajo W, como el que se acaba de describir, el sistema de flujo de trabajo automatiza el proceso mediante la creación de instancias (también conocidos como `casos` en la literatura técnica y científica de flujos de trabajo). Cada instancia es una transacción que determina un estado privado, solicitado por otras instancias del mismo proceso.

Algunos trabajos de investigación proponen la relación entre sistemas de flujos de trabajo, cómputo ubicuo y servicios [Li et al., 2008]. El modelo uFlow [Han et al., 2006] es

un *framework* basado en uWDL[6] (Lenguaje de Descripción de Flujos de Trabajo Ubicuos) que integra, administra y ejecuta servicios en entornos ubicuos.

Otros trabajos de investigación, vinculados con el cómputo ubicuo, están orientados a la administración de la información y al soporte de la colaboración. En un entorno hospitalario, la colaboración es esencial debido al trabajo en equipo de los diferentes campos de experiencia de los trabajadores, quienes no tienen una estación de trabajo fija y están en constante movimiento e interación con sus colegas o pacientes. En este ámbito, algunas aplicaciones tienen el propósito de organizar los servicios y datos de cada trabajador en términos de actividades, las cuales son mostradas en un PDA [Bardram and Henrik, 2007].

Otra propuesta de solución para el apoyo al trabajo hospitalario ofrece una aplicación consciente de contexto [Tentori and Favela, 2008], i.e., las actividades del trabajador se basan en el entorno actual donde se encuentra. La solución de esta propuesta está basada en la utilización de variables de contexto, tales como ubicación, rol y actividad.

Resulta evidente que en un entorno organizacional o en una institución de cualquier índole, el intercambio de recursos entre usuarios es de suma importancia. Sin embargo, los trabajos propuestos no han abordado la gestión de flujos de trabajo en un entorno distribuido basado en áreas, donde la administración de flujos de trabajos, asignados a los usuarios, se lleva a cabo de manera autónoma e independiente.

1.2 Planteamiento del problema

En la actualidad, la mayoría de las organizaciones cuentan con sistemas automatizados que gestionan algunos de los distintos procesos que realizan. Este tipo de sistemas administra grandes cantidades de información y permite el intercambio de datos entre las áreas de la organización. Sin embargo, la mayoría de los procesos requiere la colaboración de varios miembros de la organización para la realización de determinadas tareas. Por esta razón, surge la tecnología de flujos de trabajo, considerada como una categoría de los sistemas colaborativos [Swenson and Kent, 1995]. El objetivo de esta tecnología es proporcionar un entorno automatizado para apoyar de manera efectiva el trabajo a realizar en cualquier organización.

Los sistemas de flujos de trabajo abarcan desde la especificación formal de los procesos hasta la monitorización y ejecución de los mismos, e.g., un proceso de negocio involucra a varias personas localizadas en diferentes partes de la organización, donde cada una cuenta con aplicaciones especializadas para llevar a cabo sus tareas (ver Figura 1.3). Estos sistemas no sustituyen las tareas, sino que las adjunta en un proceso coordinado. Considere como ejemplo un proceso encargado de llevar a cabo un informe de gastos con la finalidad de ilustrar la automatización de las tareas del proceso, las cuales se suscitan de la siguiente manera: primeramente, el sistema informa que el proceso consiste en un informe de gastos (ver Figura 1.3). El encargado de este proceso procede a llenar el formulario del informe (ver Figura 1.3 ref # 2). A continuación, se invoca de manera automática la aplicación para realizar el cálculo de los gastos efectuados (ver Figura 1.3 ref. # 3). El informe de gastos es enviado a la lista de tareas del jefe de gastos para ser aprobado mediante una firma electrónica (ver Figura 1.3 ref. # 4). Posteriormente, se

[6] *Ubiquitous Workflow Description Language*

envía el informe de gastos a la lista de tareas del agente de nóminas (ver Figura 1.3 ref. # 5 y # 6) para ser autorizado (ver Figura 1.3 ref. # 7). Finalmente, el interesado recoge el cheque cuando lo requiera (ver Figura 1.3 # 8).

Figura 1.3: *Workflows* para el proceso "informe de gastos"

Los sistemas de información no cubren en su totalidad todos los procesos que engloba una organización. El gran avance tecnológico, que se tiene hoy en día, ofrece a las organizaciones la posibilidad de satisfacer la mayoría de sus necesidades y de lograr un nivel de administración más competitivo.

Nuestra propuesta consiste en ofrecer una administración basada en áreas de una organización, que controle las actividades de los usuarios nómadas. Cada área es definida como una entidad autónoma, que está encargada de la administración de su propia información, servicios, roles y flujos de trabajo. La tecnología de flujos de trabajo para sistemas de información junto con la administración de flujos de trabajo basada en áreas que se propone en este trabajo de investigación, permite una automatización más completa de los procesos que se gestionan dentro de la organización.

Si una organización carece de un soporte para la administración de las actividades de los usuarios nómadas dentro de la organización provocaría que estos:

- no tengan la certeza de lograr los objetivos de las actividades a realizar;
- desconozcan sus obligaciones y derechos, los cuales pueden cambiar dependiendo del lugar donde se encuentren; y

- violen las políticas de seguridad de la organización.

Es importante tener en cuenta que las actividades, permisos, áreas autónomas y roles de los usuarios pueden cambiar si sus necesidades cambian. Para hacer frente a este problema, la pregunta obligada es ¿Quién debería de encargarse del control de las actividades que llevan a cabo los usuarios y de los cambios de estas actividades, si se requieren? ¿El sistema? ¿El administrador del sistema? ¿Un usuario? o ¿Un grupo de usuarios?

El presente trabajo de tesis de maestría toma tales preguntas como punto de partida, definiendo así los parámetros que influirán en la administración de los flujos de trabajo de los usuarios, necesarios para el cumplimiento de sus tareas dentro de la organización.

1.3 Objetivos del proyecto

Objetivo general

Diseñar e implementar un sistema de información distribuido para administrar los flujos de trabajos de usuarios nómadas que interactúan dentro de áreas autónomas.

Objetivos específicos

Los objetivos específicos de esta tesis de maestría son los siguientes:

1. diseñar e implementar un módulo que informe al usuario nómada, en su dispositivo móvil, la lista de actividades o tareas que debe llevar a cabo dentro de una área autónoma;

2. definir la estructura lógica de cada una de las areas autónomas que componen una organización;

3. diseñar y construir la base de datos que almacena la información correspondiente del área:

   (a) contexto del área;

   (b) perfil del usuario nómada; y

   (c) registro de las actividades y/o tareas de los usuarios nómadas.

4. crear las funciones de administración del flujo de trabajo, e.g., agregar actividad, eliminar actividad, asignar actividad particular, ejecutar actividad y determinar actividad actual;

5. definir un mecanismo que verifique las condiciones temporales Rol-Usuario y Permisos-Rol asignados al usuario nómada; y

6. definir una interfaz que permita obtener información cuando un usuario nómada cambie de área autónoma e.g., su nueva lista de actividades, recursos y servicios.

1.4 Organización del documento

El presente documento de tesis está organizado en cinco capítulos (ver figura 1.4). Después de haber presentado el contexto de investigación donde se ubica la presente tesis y de haber planteado el problema que se pretende resolver, así como el objetivo general y los objetivos específicos que se persiguen, en el capítulo 2 se describe el estado del arte.

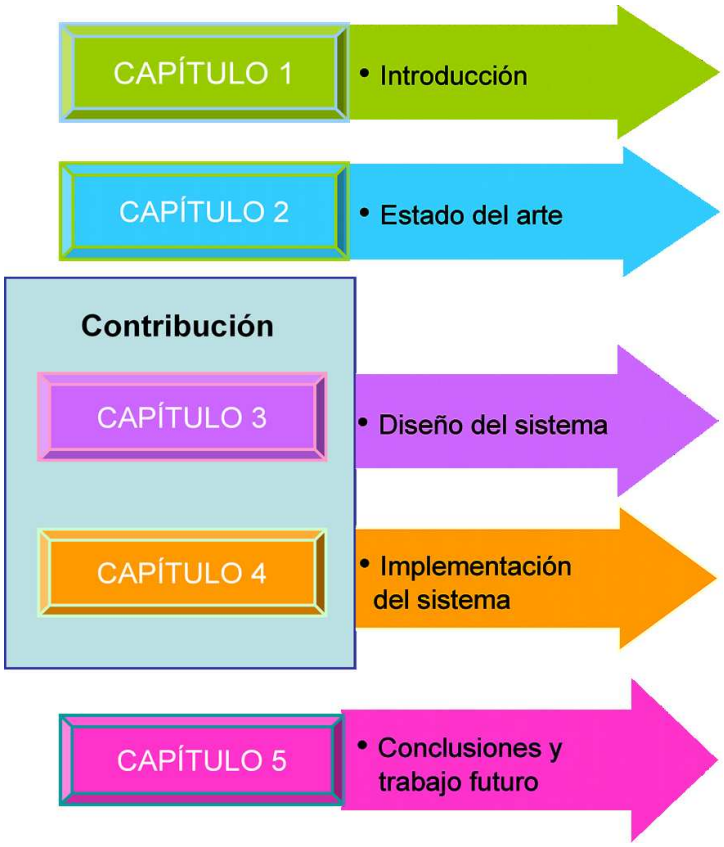

Figura 1.4: Organización del documento de tesis

Particularmente, se analizan los principales trabajos de investigación referentes al modelo RBAC y a algunas aplicaciones conscientes de contexto y de actividades, las cuales han sido utilizadas en entornos reales (e.g., hospitales e instituciones) para proveer información, servicios y actividades a los usuarios según su ámbito actual (e.g., localización).

En el capítulo 3 se detalla el diseño del sistema W-RBACSoft [7] , el cual lleva a cabo la administración de flujos de trabajo en un entorno distribuido. W-RBACSoft se encarga de la administración de actividades, recursos e información de las áreas autónomas. En primer lugar, se presenta un escenario que expone la utilidad del sistema propuesto en entornos organizacionales. A continuación, se describe la arquitectura del sistema, los módulos que lo componen, así como la estructura y organización de las áreas para proporcionar un adecuado entorno distribuido a los usuarios nómadas.

En el capítulo 4 se detalla la implementación de algunos módulos del sistema W-RBACSoft. Se describe la arquitectura funcional del sistema y se analizan distintos casos

[7] Workflow Software - Role Based Access Control

de estudio que involucran usuarios con diferentes objetivos y estatus. Adicionalmente, se detallan las reglas que sigue el motor de flujos de trabajo para su gestión.

Finalmente, en el Capítulo 5, se presentan las conclusiones de este trabajo de investigación así como algunas posibles prolongaciones a futuro.

Capítulo 2

Estado del Arte

El objetivo del presente capítulo es realizar un estudio de los trabajos relacionados con el contexto de investigación, donde se inscribe el presente trabajo de tesis. En primer lugar se describe el modelo RBAC, considerado como el arquetipo de base para la construcción de sistemas de control de acceso. Asimismo, se detallan los sub-modelos descendientes de RBAC, los cuales fueron concebidos en distintos ámbitos de estudio tales como: control de acceso en organizaciones, entornos distribuidos y flujos de trabajo (cf. sección 2.1). Posteriormente, se describe el sistema de flujos de trabajo Uflow, así como algunas de las propuestas más importantes que están relacionadas con el sistema que se sugiere en esta tesis. Estas propuestas tienen como objetivo dar apoyo a las actividades que realizan los trabajadores de un hospital, sin embargo cada una propone su propias técnicas de solución (cf. sección 2.2). Finalmente, se presenta un estudio comparativo de los sistemas analizados, con el fin de poner en evidencia la aportación que ofrece nuestro sistema (cf. sección 2.3).

2.1 Modelo RBAC

El concepto de control de acceso basado a roles (RBAC) surgió a principios de los setentas. Los pioneros del modelo RBAC fueron los sistemas en línea por tratarse de sistemas multi-usuarios, i.e., sistemas encargados de atender a varios usuarios. El objetivo principal de RBAC es asociar un determinado rol a un usuario y posteriormente asignarle permisos de acuerdo con este rol.

Los roles cuentan con las siguientes características [Ravi et al., 1996]:

- controlan las funciones de una organización;
- suelen ser asignados con base en las responsabilidades de los usuarios y pueden ser reasignados de un usuario a otro; y
- son estables debido a que las funciones de una organización cambian con poca frecuencia.

El modelo RBAC está descrito por (ver Figura 2.1): a) las entidades `usuario`, `rol` y `privilegio`, b) las relaciones `juega`, `hereda`, `tiene` e `implica` entre estas entidades y c) restricciones impuestas a dichas relaciones.

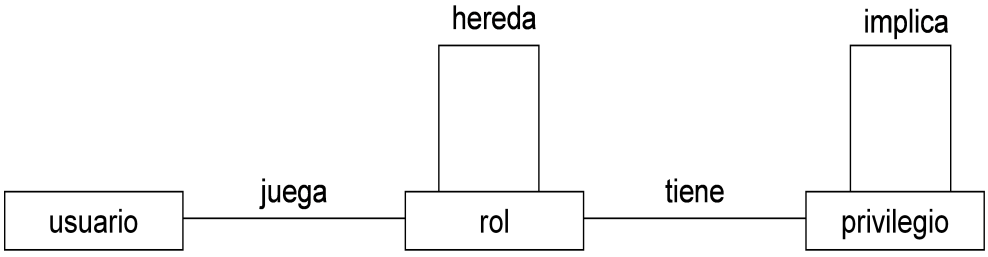

Figura 2.1: Modelo RBAC

Las entidades del modelo RBAC están definidas de la siguiente manera. Sea U el conjunto de usuarios, R el conjunto de roles y P el conjunto de privilegios:

- una entidad `usuario` $u \in U$ representa a un usuario individual;
- un `privilegio` $p \in P$ define clases de derechos para ejecutar operaciones, tareas y acceso de datos; y
- un `rol` $r \in R$ denota un grupo de privilegios o habilidades que puede ser asignado a los usuarios.

Las relaciones del modelo RBAC están definidas de la siguiente manera:

- `juega` $(u,\ r)$, tal que $u \in U$, $r \in R$, significa que el usuario u juega el rol r;
- `hereda` $(r_1,\ r_2)$, tal que $r_1,\ r_2 \in R$, supone que el rol r_1 es una clase de r_2 (r_1 hereda todos los privilegios de r_2);
- `tiene` $(r,\ p)$, tal que $r \in R$, $p \in P$, señala que el rol r tiene el privilegio p; y
- `implica` $(p_1,\ p_2)$, tal que $p_1,\ p_2 \in P$, indica que el estado del privilegio p_1 es más importante que el estado del privilegio p_2, por lo tanto p_1 reemplaza o incluye a p_2.

2.1.1 Submodelo Or-BAC

Las políticas de seguridad de los sistemas de información actuales son cada vez más complejas y difíciles de gestionar, ya que manejan comunidades virtuales [Kalam et al., 2003]. La finalidad de una comunidad virtual es administrar organizaciones heterogéneas e independientes, cuyas interacciones tienen lugar en un espacio virtual como Internet [Cuppens and Miége, 2003]. Las diferentes organizaciones que participan en una comunidad virtual deben ser capaces de definir sus propias políticas de seguridad en términos de privilegios (permisos, obligaciones o limitaciones) para poder controlar las actividades de sus miembros. Dichas políticas deben adaptarse a los requerimientos de las diversas organizaciones participantes en función de su contexto, e.g., condiciones temporales, espaciales y comportamentales.

El objetivo principal del modelo Or-BAC es facilitar la especificación de las políticas de seguridad de una organización para dirigir las acciones realizadas por los usuarios

sobre los objetos. Or-BAC asigna permisos a un usuario con base en su rol y en la vista (una determinada aplicación que forma parte del sistema) que requiere acceder. Por ejemplo, considere la siguiente política "*al rol* `médico` *se le asigna el permiso de llevar a cabo la actividad* `consulta` *en la vista* `registro médico`". En este ejemplo, se ilustra una política estática porque una vez definida permanecerá sin cambios. Sin embargo, el modelo Or-BAC también permite especificar políticas de manera dinámica en función del contexto actual del usuario. Por lo tanto, es importante considerar qué tipo de información debe gestionar un determinado sistema de información para hacer frente a las condiciones contextuales que se tienen contempladas.

Las entidades del modelo Or-RBAC están definidas de la siguiente manera: sea Org el conjunto de organizaciónes, U el conjunto de usuarios, A el conjunto de acciones, O el conjunto de objetos, R el conjunto de roles, T el conjunto de actividades, V el conjunto de vistas y C el conjunto de contextos. Las políticas de seguridad del modelo Or-RBAC están representadas por el siguiente conjunto de reglas de hechos:

- `permite`(u, a, o)
- `debe`(u, a, o)
- `limita`(u, a, o)

Por ejemplo, la primera regla de hechos implica que un usuario $u \in$ U (e.g, *Rafael*) tiene permitido realizar una acción $a \in$ A (e.g., *leer*) en un objeto $o \in$ O (e.g., *expediente médico de Raúl*):

`permite`(*Rafael, leer, expediente médico de Raúl*)

Sin embargo, la creación de todas las reglas de hechos necesarias implica una tediosa y difícil gestión de tareas, ya que cada vez que se crea un nuevo usuario, objeto o acción, es necesario introducir nuevas reglas que especifiquen los privilegios correspondientes. Para facilitar la creación de políticas de seguridad, las reglas de hechos pueden ser representadas de la siguiente manera [Nicodemos et al., 2001]:

- $\forall u, \forall a, \forall o$ (`condición` $\rightarrow$ `permite`(u, a, o))
- $\forall u, \forall a, \forall o$ (`condición` $\rightarrow$ `debe`(u, a, o))
- $\forall u, \forall a, \forall o$ (`condición` $\rightarrow$ `limita`(u, a, o))

Si la `condición` es satisfactoria, el usuario u está autorizado a realizar la acción a sobre el objeto o, $\forall u \in$ U, $\forall a \in$ A y $\forall o \in$ O. La estructura condicional de las reglas anteriores está basada en la siguiente sintaxis:

- `condUsuario`(u) $\wedge$ `condAcción`(a) $\wedge$ `condObjeto`(o) $\wedge$ `restricción`(u, a, o)

Donde `condUsuario`(u), `condAcción`(a) y `condObjeto`(o) son respectivamente las condiciones del usuario u, de la acción a y del objeto o, las cuales deben cumplirse de manera separada. Por su parte, `restricción`(u, a, o) es una condición adicional que es necesario satisfacer para que el usuario u lleve a cabo la acción a en el objeto o. Por ejemplo, considere la política siguiente: *"al rol* `médico` *se le permite* `consultar` *el* `registro médico` *de sus pacientes"*. Con base en la sintaxis anterior, la regla de hechos quedaría definida así:

`restricción`(*médico, consultar, registro médico*)

En este ejemplo, la restricción determina que la acción *consultar* se lleva a cabo si el objeto *registro médico* corresponde al paciente del *médico*.

Las entidades `usuario`, `acción` y `objeto` del modelo RBAC son representados respectivamente por las entidades `rol`, `actividad` y `vista` en el modelo Or-BAC, donde:

- un `rol` determina una función que se desarrolla dentro de la organización;
- una `actividad` reagrupa las acciones que comparten los mismos principios; y
- una `vista` corresponde al conjunto de objetos que cumplen con una prioridad común;

Cualquier entidad del modelo Or-BAC puede tener atributos, los cuales son representados por funciones asociadas a la entidad, e.g., si u es un usuario, entonces `nombre`(u) y `dirección`(u) representan respectivamente el `nombre` y la `dirección` del usuario u.

El modelo Or-BAC ofrece las siguientes relaciones:

- `otorga` es una relación sobre el dominio Org $\times$ U $\times$ R

La relación `otorga`(*org, u, r*), donde *org* $\in$ Org, $u \in$ U y $r \in$ R, significa que la organización *org* otorga al usuario u el rol r. Está relación permite modelar situaciones donde un usuario juegue varios roles, pero en diferentes organizaciones.

- `usa` es una relación sobre el dominio Org $\times$ O $\times$ V

La relación `usa`(*org, o, v)*, donde *org* $\in$ Org, $o \in$ O y $v \in$ V, denota que la organización *org* utiliza objetos o en la vista v. Esta relación es útil para caracterizar organizaciones que tienen diferentes definiciones de una vista. Por ejemplo, la vista *regisro médico* está definida en el hospital *Los Angeles* como un conjunto de documentos de texto, mientras que en el hospital *La Raza* como un conjunto de tuplas de una base de datos relacional.

- `considera` es una relación sobre el dominio Org $\times$ A $\times$ T

La relación `considera`(*org, a, t*), donde *org* $\in$ Org, $a \in$ A y $t \in$ T, designa que la organización *org* considera la acción a dentro de la actividad t. Esta relación permite que diversas organizaciones puedan definir una misma acción bajo distintas actividades o diversas acciones bajo la misma actividad. Por ejemplo, dada la actividad *consulta*, en el

hospital *Los Ángeles*, la acción es *leer* ya que cuenta con un fichero de datos, mientras que en el hospital *La Raza*, la acción es *seleccionar* porque utiliza una base de datos relacional.

En Or-BAC estas condiciones son modeladas bajo el concepto de contexto, el cual tiene un nombre y una definición que depende de la organización, e.g., el contexto *cuidado médico* está englobado en varias circunstancias, tales como *urgencias*, *investigación médica* y *asistencia médica*. La regla que engloba los conceptos de contexto y entidades (organización, usuario, objeto y acción) está definida por la relación `define`:

- `define` es una relación sobre el dominio Org × U × A × O × C

La relación `define`(*org*, *u*, *a*, *o*, *c*), donde *org* ∈ Org, *u* ∈ U, *a* ∈ A, *o* ∈ O y *c* ∈ C, señala que dentro de la organización *org*, el contexto *c* contiene un usuario *u*, una acción *a* y un objeto *o*. Las condiciones necesarias para que un contexto dado se vincule con usuarios, objetos y acciones dentro de una organización determinada, deberán ser especificadas formalmente por reglas lógicas. Por ejemplo, en el hospital *Los Ángeles*, el contexto *asistencia médica* involucra al usuario *u*, a la acción *a* y al objeto *o*, si y solo si el *registro médico* (`nombre`(*o*)) pertenece al paciente del usuario *u* (`paciente`(*u*))):

- `define`(*Los ángeles*, *u*, *a*, *o*, *asistencia médica*) ↔ `nombre`(*o*) ∈ `paciente`(*u*)) ∀*u* ∈ U, ∀*a* ∈ A, ∀*o* ∈ O

De esta manera, se lleva a cabo la construcción de reglas lógicas enfocadas al contexto de la organización. Estas reglas expresan los diferentes tipos de condiciones y restricciones que controlan su activación en las políticas de seguridad de la organización.

2.1.2 Submodelos de W-RBAC

El modelo W-RBAC agrega al modelo RBAC nuevas entidades para abordar el dominio de los flujos de trabajo. Su objetivo es facilitar la asignación permisos con base en las tareas y en el rol del usuario [Wainer et al., 2003]. Del modelo W-RBAC surgen los submodelos W0-RBAC y W1-RBAC.

W0-RBAC

El modelo W0-RBAC agrega al modelo RBAC las entidades `caso` y `unidad organizacional`, así como las relaciones `engloba, miembro, líder` y `hacedor` (ver Figura 2.2).

Entidad `caso`

El objetivo de esta entidad es prohibir que un mismo usuario lleve a cabo dos actividades al mismo tiempo, e.g., un usuario no puede ser piloto y azafata en un mismo vuelo. Estas restricciones en particular pueden ser capturadas en el modelo RBAC mediante el uso de sesiones, que implican la asociación entre un usuario y un rol en algún intervalo de tiempo. En el ejemplo anterior, se puede prohibir la sesión si un usuario está vinculado simultáneamente a los roles de piloto y azafata.

En aplicaciones de flujos de trabajo, es muy importante la culminación de los procesos de una organización dado que su objetivo es automatizarlos. Por lo general, los miembros

que integran la organización, se encargan de realizar actividades determinadas en cada uno de los procesos. Sin embargo, es importante mantener el control de cada una de dichas actividades para no violar las políticas de seguridad de la organización. Por ejemplo, se debe prohibir que un mismo usuario ejecute las actividades de solicitud y aprobación en un proceso de reembolso.

Para ilustrar este requerimiento, suponga que se tienen dos procesos. En el primer proceso, Guadalupe está encargada de aprobar o rechazar la solicitud de reembolso de Carolina y en el segundo proceso Guadalupe hace una solicitud de reembolso, pero el encargado de aprobarla o rechazarla es su jefe Armando. En caso de que Guadalupe aprobara su propia solicitud en el segundo proceso, estaría violando una política de seguridad. En este ejemplo, el concepto de sesión es inadecuado porque Guadalupe puede jugar dos roles al mismo tiempo, los cuales permiten llevar a cabo actividades diferentes: 1) solicitar un reembolso y 2) aprobar o rechazar una solicitud de reembolso, respectivamente. Sin embargo, es aceptable activar ambos roles porque estos se juegan en distintos procesos.

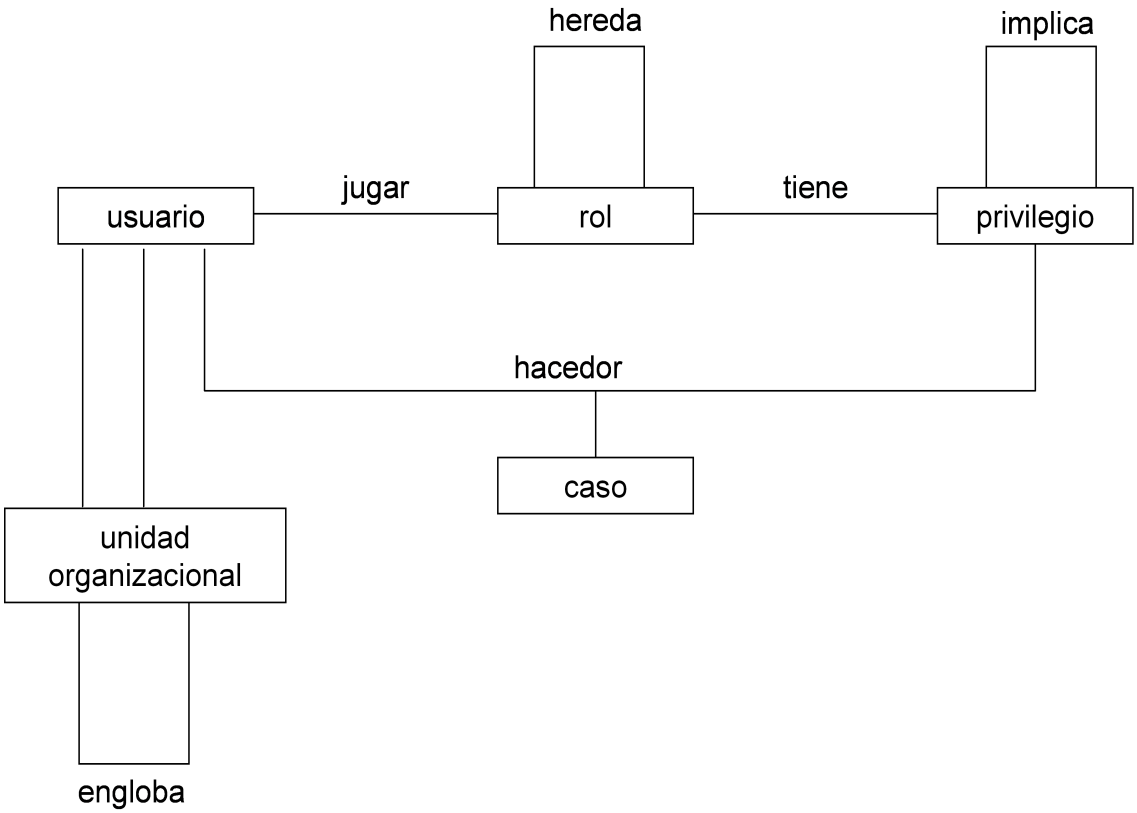

Figura 2.2: Modelo W0-RBAC

La entidad `caso` y la relación `hacedor` (*doer*) permiten hacer referencia a la instancia de un proceso que, como se vio en el ejemplo anterior, es de suma importancia:

- `hacedor` es una relación sobre el dominio U × P × C

La relación `hacedor`(*u, p, c*), donde $u \in$ U, $p \in$ P, $c \in$ C, significa que el usuario u tiene el privilegio p en el contexto c. Si está relación se asocia a una actividad del usuario, en vez de un privilegio, quedaría como `hacedor`(*u, t, c*), la cual establece que el usuario u ejecuta la actividad t en el contexto c.

`Entidad unidad organizacional`

Dentro de las organizaciones, el concepto de rol es común. Las personas tienen movilidad entre departamentos, divisiones o grupos, así como diversos jefes debido a los distintos niveles jerárquicos que posee una organización.

La entidad `unidad organizacional` ofrece la relación `hereda`, la cual está definida como una relación de herencia tal como su nombre lo indica. La expreción `hereda`(*programador-c, programador*), especifica que un *programador-c* tiene todos los privilegios de un *programador*, i.e., los niveles más altos en la organización acumulan los privilegios de los que están bajo su mando. La propuesta de `unidad organizacional` es vista desde la perspectiva de un modelo basado en jerarquías que hace frente a la asignación tanto estática como dinámica de usuarios a grupos, departamentos y divisiones.

Las relaciones que definen a la entidad `unidad organizacional` son las siguientes:

- `engloba`(d_1, d_2), donde d_1, $d_2 \in OU$, significa que la unidad organizacional d_1 incluye la unidad de organización d_2, e.g., `engloba`(*Departamento de Ingeniería*, *equipo del proyecto 12*) indica que el *equipo del proyecto 12* es parte del *Departamento de Ingeniería*;

- `miembro`(*u, d*), donde $u \in u$, $d \in OU$, señala que el usuario u es miembro de la unidad organizacional d; y

- `líder`(*u, d*), donde $u \in U$, supone que el usuario u es el líder o el responsable de la unidad organizacional d.

W1-RBAC

Según en el modelo W0-RBAC, las políticas de seguridad requeridas para llevar a cabo un proceso se basan en determinar quién y en qué momento se ejecutan las actividades que lo componen, sin permitir cambios en las políticas aunque sea necesario. Por el contrario, el modelo W1-RBAC permite realizar cambios temporales en las políticas, ya que agrega el manejo de excepciones al modelo W0-RBAC.

El modelo W1-RBAC se encarga de asignar prioridades a las restricciones, de acuerdo con el nivel de importancia que estas tengan. Es claro que algunas restricciones son más importantes que otras y que diferentes situaciones pueden permitir diferentes grados de violación de políticas. En general, algunas restricciones son más importantes que otras, por esta razón las menos importantes pueden ser eliminadas.

W1-RBAC incorpora excepciones a la gestión de flujos de trabajo. En la literatura de flujos de trabajo se reconoce que, en situaciones reales, a menudo las actividades especificadas en el flujo de trabajo de un proceso pueden ser violadas con la finalidad de que concluya dicho proceso. A la violación de estas actividades se le conoce como manejo de excepciones. Por ejemplo, suponga que un cliente muy importante está llevando a cabo

un proceso, pero tiene prisa por concluirlo. Por esta razón, el orden de ejecución de las actividades constituyentes puede cambiar o bien distintas personas que estén disponibles pueden llevar a cabo dichas actividades. Las excepciones manejan acciones. Por lo tanto, una excepción importante en los sistemas de flujos de trabajo es la asignación de actividades a usuarios, que usualmente no están autorizados a realizarlas.

Por ejemplo, una de las políticas de una organización puede exigir que las actividades de recepción de la consulta de un cliente y su respectiva respuesta deban ser realizadas por la misma persona, para dar al cliente una atención personal. Suponga que Juan recibió una consulta técnica de Kensington Corp., que es un cliente muy importante. El proceso encargado de construir la respuesta procede normalmente, pero Juan se encuentra ausente en el momento en el que la respuesta tiene que ser enviada al cliente. En este caso, es más importante responder al cliente que esperar a que Juan regrese para que de la respuesta personalmente. Por esta razón, otra persona debe ponerse en contacto con el cliente para atenderlo, violando las restricciones entre las actividades de recibir y responder la consulta de un cliente sin importar quién las lleve a cabo.

2.2 *Framework* uFlow

Los sistemas de administración de flujos de trabajo (WfMS[1]) apoyan la cooperación y coordinación entre los colaboradores para la realización de las tareas de una organización, las cuales suelen ser modeladas como procesos de flujos de trabajo [Gwan et al., 2003]. El modelado de un flujo de trabajo es la representación computarizada de los procesos que se llevan a cabo en una organización. Particularmente, un flujo de trabajo definen las condiciones de inicio y fin de cada proceso, las actividades de cada proceso y el control del flujo de datos entre dichas actividades. Una actividad es un paso lógico, dentro de un flujo de trabajo, que está compuesta por: a) la información de arranque y parada, b) el número de usuarios que pueden participar, c) las herramientas y/o los datos necesarios para realizar la actividad y d) las restricciones que especifican cómo la actividad debe ser completada.

uFlow es un *framework* que permite crear un entorno ubicuo capaz de proporcionar servicios apropiados a los usuarios con base en su situación [Han et al., 2006]. uFlow está constituido por un modelo de contexto estructural y un lenguaje de descripción de flujos de trabajo ubicuos denominado uWDL[2]. El modelo estructural expresa la información contextual mediante el lenguaje uWDL, el cual también permite describir los servicios disponibles. uFlow emplea la tecnología de los servicios Web, con el fin de ofrecer a los usuarios los servicios requeridos por su situación actual.

En entornos de computación ubicua, el contexto representa la información que se utiliza para caracterizar la situación de una entidad. uWDL proporciona funcionalidades para seleccionar un servicio apropiado en función del contexto, así como de los perfiles y eventos provenientes de varias fuentes. Una vez que se tiene la información correspondiente al contexto del usuario, el editor de escenario (*uWDL scenario editor*) crea un documento uWDL (*uWDL scenario document*), el cual contiene el escenario de los servicios

[1] *Workflows - Management Systems*
[2] *ubiquitous Workflow Description Language*

disponibles en el flujo de trabajo.

El manejador uWDL consiste de un analizador sintáctico (*uWDL Parser*) y un enlazador (*Context-Mapper*) (ver Figura 2.3). El analizador sintáctico verifica dicho documento uWDL y produce un documento DIAST[3], el cual representa la información como un árbol que define la transición de servicios en el flujo de trabajo.

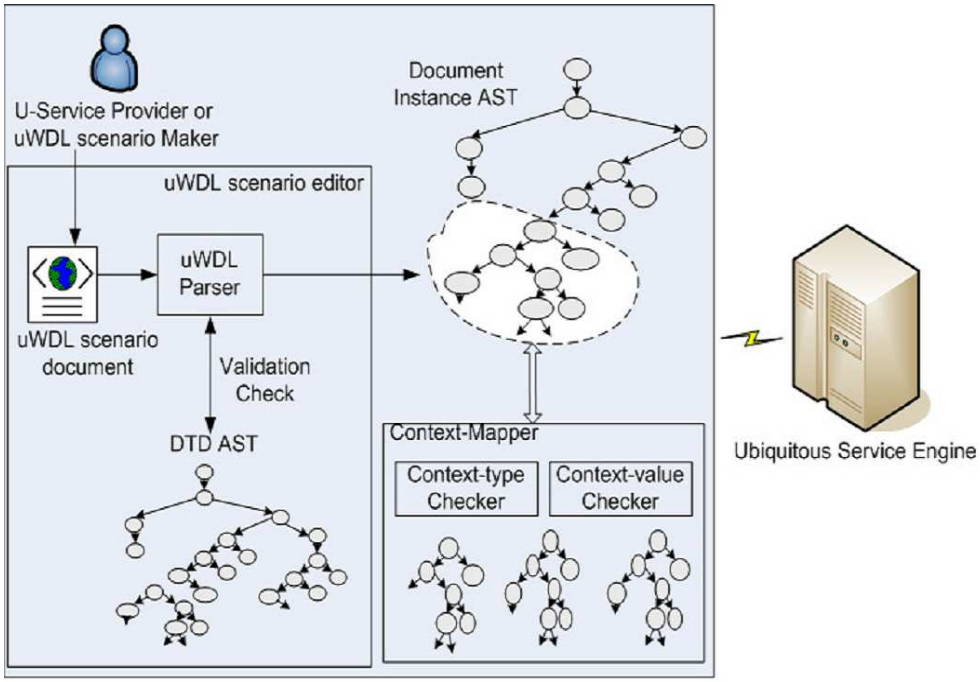

Figura 2.3: Arquitectura del manejador uWDL para el tratamiento del contexto

La información contextual del usuario es obtenida por medio de una red de sensores. Esta información puede estar embebida y definida como una entidad que consiste de un usuario, una acción y un objeto. El enlazador de contexto (*Context-Mapper*) extrae los tipos de contexto (*Context-type Checker*) y los valores (*Context-value Checker*) de la entidad objetivo. Posteriormente, el enlazador compara esta información con el árbol del documento DIAST para encontrar la entidad que coincida o empate y así definir los servicios a otorgar.

Por ejemplo, suponga que Juan va a realizar una presentación en la sala 313 a las 10:00 A.M. Cuando Juan se traslada a dicha sala, un sensor que se encuentra en la puerta obtiene su información básica (e.g., nombre y dirección IP de *laptop*) y la almacena en el servidor. Si las condiciones actuales, tales como la ubicación de Juan, su situación (e.g., su perfil de conferencista) y el momento actual satisfacen los contextos descritos en el escenario de los servicios del flujo de trabajo, entonces el servidor descargará el archivo de su presentación y lo ejecutará.

El desarrollador de aplicaciones puede emplear entidades en el escenario uWDL, tales como usuario, acción y objeto, para decidir qué servicio será seleccionado con base en el contexto del usuario (ver Figura 2.4). En el lenguaje uWDL, el escenario descrito consiste

[3] *Document Instance Abstract Syntax Tree*

en un número limitado de contextos ya definidos, sin embargo la red de sensores puede producir innumerables situaciones que pueden coincidir con el contexto de cada usuario.

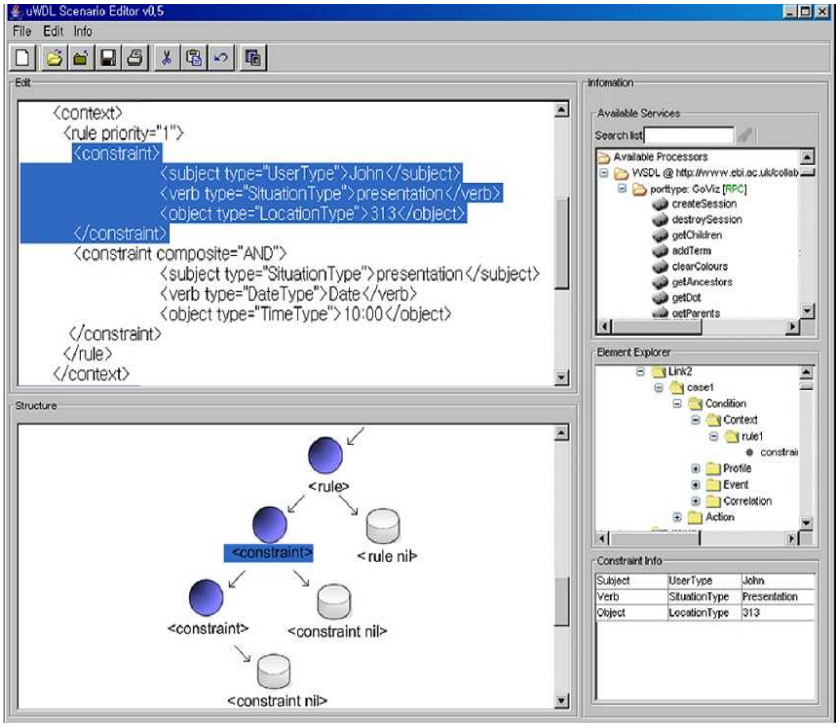

Figura 2.4: Sub-arbol DIAST analizado sintácticamente por el lenguaje uWDL

2.3 Cómputo ubicuo para el soporte de hospitales

Generalmente, el entorno de trabajo de los hospitales está orientado al trabajo en equipo ya que, de una manera constante, se lleva a cabo la colaboración entre los diferentes campos de experiencia de los trabajadores. Por esta razón, esta forma de trabajo es conocida como "trabajo nómada", debido a que las personas están en constante movimiento e interacción con sus colegas o pacientes.

Algunos trabajos de investigación toman en cuenta los aspectos mencionados anteriormente para la implementación del cómputo ubicuo en entornos hospitalarios. Los sistemas creados para este tipo de entornos se centran principalmente en la administración de información, en el soporte de actividades y en la colaboración entre trabajadores. Estos sistemas permiten organizar los servicios y datos de los trabajadores en términos de actividades, i.e., el entorno dederá ser capaz de determinar qué actividad le corresponde llevar a cabo a cada trabajador, dependiendo del lugar donde se encuentre. Cada uno de los trabajos de investigación que se describe en las secciones siguientes propone una solución distinta. La primera propuesta define una solución basada en la conciencia de actividades, mientras que la segunda propuesta está basada en la conciencia de contexto.

2.3.1 Conciencia de actividades

Las clínicas y los hospitales manejan grandes volúmenes de datos compartidos, tales como registros de pacientes y rayos X. El trabajo que se presenta en este entorno está orientado a la colaboración entre personas, ya que las actividades implican hablar con los pacientes, atender conferencias e intercambiar opiniones entre colegas. Los médicos y las enfermeras usualmente no tienen una estación de trabajo fija, ya que la naturaleza de su trabajo hace que estén en constante movimiento. El proceso de este trabajo es largo debido a que se tiene que recolectar información proveniente de varios lugares.

Por ejemplo, suponga que un médico se dirige al cuarto de uno de sus pacientes, despúes se para frente a su cama e ingresa notas sobre su estado de salud en su expediente eléctronico (EPR); posteriormente, se reúne con sus colegas radiólogos para analizar los resultados de la radiología, así como el tratamiento apropiado para su paciente; durante esta reunión, el médico consulta los catálogos de medicamentos; más tarde, el laboratorio le envía los resultados de las pruebas de sangre que se le practicaron a su paciente [Bardram and Henrik, 2007]. Este ejemplo muestra la necesidad de que los médicos accedan a varias aplicaciones que muestren información relevante de forma adecuada.

Para hacer frente a este requerimiento, el *framework* ABC implementa el "cómputo basado en actividades". ABC facilita la organización de las aplicaciones a las que el usuario debe acceder. Los datos y servicios son definidos en términos de actividades, las cuales pueden ser recreadas y compartidas. ABC permite que los usuarios cambien fácilmente de contexto mediante cualquier dispositivo que tengan a la mano.

El *framework* ABC se enfoca en cuatro temas médicos que reflejan las áreas de trabajo más importantes de los hospitales, tales como:

- administración de medicamentos por enfermeras;
- conferencias médicas;
- sugerencias médicas; y
- colaboración entre colegas.

El aspecto más importante en ABC es el soporte de la colaboración entre los trabajadores. Por esta razón, permite que los usuarios compartan aplicaciones asíncronas o síncronas. Una aplicación asíncrona permite compartir la actividad en curso para que otros usuarios la observen. Por su parte, una aplicación síncrona permite que los colaboradores interactúen simultáneamente para intercambiar puntos de vista sobre un caso específico.

La Figura 2.5 ilustra algunas de las actividades que se realizan dentro de un entorno hospitalario, tales como: a) un médico lleva a cabo la actividad de descubrimiento del medicamento del paciente, b) un radiólogo comparte los resultados de los exámenes de un paciente con sus colegas para definir un diagnóstico y c) un médico inicia una actividad a realizar, la muestra en la pantalla de su dispositivo y la comparte con sus colegas.

El *framework* ABC permite pausar actividades y reiniciarlas más tarde. Por ejemplo, si un médico está prescribiendo el medicamento al paciente A y al mismo tiempo está viendo el registro médico del paciente B, ambas actividades podrían provocar una confusión tanto

Figura 2.5: Colaboración entre colegas para obtener un determinado diagnóstico

para el médico como para el sistema. Por esta razón, lo más factible es detener la actividad correspondiente al paciente A y reiniciarla después de terminar la actividad del paciente B. ABC cuenta con una función llamada *roaming*, la cual permite que un colaborador pueda reiniciar una actividad en cualquier dispositivo.

La funcionalidad de este *framework* se basa en la gestión de la información, de los servicios y de los recursos, permitiendo a los usuarios identificar, crear y administrar actividades del trabajo diario que realizan. Desde el punto de vista de los diseñadores, los dispositivos con capacidades pequeñas (e.g., dispositivos móviles) no son adecuados para el uso de este *framework*, debido a que sus capacidades de procesamiento son muy bajas y sus pantallas son de tamaño inadecuado para la visualización de la información.

2.3.2 Conciencia de contexto

Los hospitales constituyen un entorno adecuado para el desarrollo de aplicaciones que soportan el trabajo colaborativo. La propuesta de solución al apoyo de las tareas que realizan los trabajadores en este entorno concierne la "computación basada en el contexto" [Favela et al., 2007].

La información necesaria para realizar el trabajo de hospitales es altamente dependiente de variables contextuales, tales como ubicación, rol, hora del día y actividad, e.g., el documento más relevante de una enfermera que se encuentra frente a un paciente es su cuadro médico, mientras que para un médico es el registro médico del paciente. Esta propuesta se enfoca precisamente en la comunicación intra-hospitalaria y en el acceso de información basada en la localización y en el rol del usuario.

El objetivo de las aplicaciones basadas en contexto es aproximarse a la estimación automática de las actividades realizadas por los trabajadores del hospital. Por ejemplo, suponga que se requiere la presencia de una enfermera en la unidad de emergencia; si el sistema estima con precisión la información necesaria para llevar a cabo la actividad, debe ser capaz de decidir qué enfermera llamar, con base en la disponibilidad de todas las enfermeras.

La tecnología utilizada para obtener esta aproximación se basa en el uso del entrenamiento de una red neuronal para asociar la información contextual, tal como la localización de los trabajadores dentro del hospital, los artefactos más utilizados, los colegas que pueden colaborar dependiendo de la actividad realizada y la hora del día para llevar a cabo esta colaboración. La información requerida para el entrenamiento de la red

neuronal, se obtuvo mediante el desarrollo de una red de sensores para monitorear las variables contextuales y así estimar las actividades de los usuarios. Esta información se capturaró durante un periodo de 8 meses, donde cinco médicos internos, cinco enfermeras y cinco médicos especialistas fueron observados durante su trabajo dentro de los dos turnos (ver Figura 2.6).

Figura 2.6: Trabajadores del hospital durante uno de los días de observación

Como se mencionó anteriormente, las actividades dependen de diversas variables contextuales, e.g., la ubicación es codificada por un número que es asignado a cada lugar del hospital, mientras que los artefactos utilizados son codificados mediante una representación binaria: 1 si el artefacto está siendo utilizado y 0 en caso contrario. De esta manera, las variables pueden ser gestionadas de varias formas para tratar la complejidad del entrenamiento de la red neuronal.

La Figura 2.7 ilustra la información reportada en un formato de observación, donde cada variable es codificada con los valores correspondientes. Por ejemplo, el formato de observación indica que la actividad IM (*Information Management*): 1) se llevará a cabo a las 8:57; 2) corresponde a la realización del diagnóstico del paciente; 3) utiliza accesorios de oficina y su respectiva hoja de trabajo y 4) finalmente, se lleva a cabo en el pasillo del hospital. Posteriormente, la información obtenida se transforma en entradas y salidas para la lectura y el entrenamiento de la red neuronal.

La Figura 2.8 ilustra la interfaz de usuario del sistema mobileSJ. En la parte superior, se muestran las aplicaciones con las que el usuario puede interactuar mientras que, en la parte inferior, se aprecian dos despliegues donde el sistema le sugiere al usuario la información necesaria para realizar su actividad en curso. Si el usuario selecciona uno de los despliegues, la aplicación correspondiente aparacerá en la pantalla.

El sistema mobileSJ sirve de apoyo a los usuarios móviles en la administración de sus múltiples actividades y colaboraciones. Esta aplicación transforma un entorno de trabajo físico en una representación computacional, i.e., el sistema mobileSJ permite al usuario gestionar sus múltiples actividades, información y recursos contextuales, mientras se encuentra lejos de su escritorio o estación de trabajo fija, ya que las aplicaciones pueden ser ejecutadas en una PDA o *smartphone*.

Entradas	TIME 8:57	LOC 3											Tiempo y ubicación
	NOBM 1	MI 0	PC 0	MB 0	EO 0	FM 0	ME 0	JP 0	TE O	AD O	OT 0		Interacción
	EM 0	AO 1	HE 1	EC 0	FA 0	RE 0	MC 0	MR 0	MM 0	CA 0	EQ 0	EQ 0	NONE 0	Artefactos
Salidas	CCA 0	PC 0	C 0	IM 1	T 0								Actividades

Hora	¿Qué actividad esta realizando?	Artefactos utilizados	Participantes	Ubicación	Actividades
8:57:30	Reporte del diagnostico del paciente	Accesorios de oficina y hojas de trabajo		Pasillo del hospital	IM

Figura 2.7: Estructura de los datos para el entrenamiento de la red neuronal

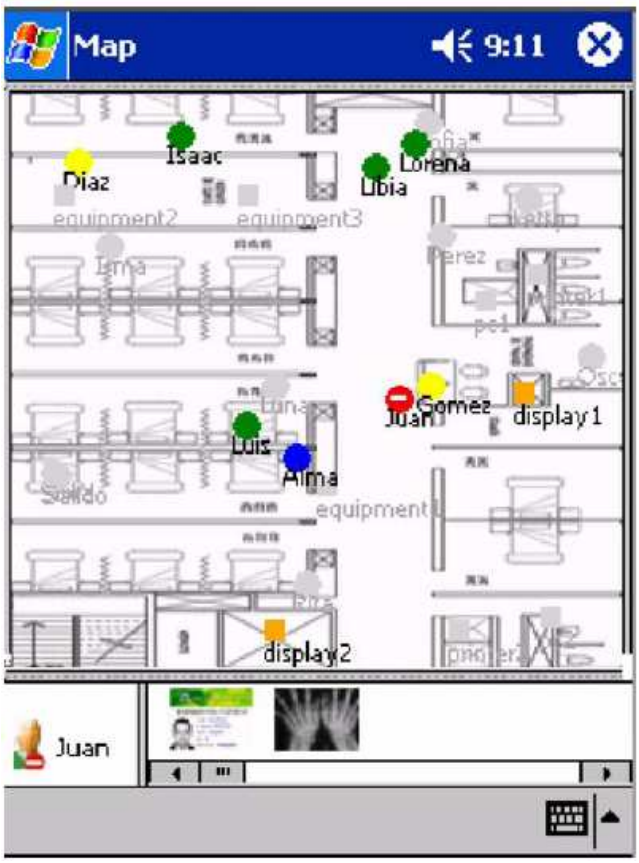

Figura 2.8: Interfaz de usuario del registro médico de un paciente

2.4 Análisis del trabajo relacionado

Los trabajos analizados en este capítulo ofrecen modelos que permiten crear políticas para el control de acceso, así como herramientas para implementar entornos ubicuos, donde los usuarios reciben información relevante de acuerdo a su contexto. El modelo RBAC es considerado como el modelo de base para la construcción de sistemas de control de acceso, ya que las políticas de seguridad se basan en definir permisos, restricciones y obligaciones de los usuarios. Este modelo ha sido adaptado a los requerimientos que exigen diversos ámbitos.

El sub-modelo Or-BAC se centra en la administración del control de acceso dentro de las organizaciones. Las diferentes organizaciones que participan en una comunidad virtual se encargan de expresar sus propias políticas de seguridad, cuyas reglas se adaptan a las condiciones temporales del usuario, e.g., rol y ubicación. El objetivo de este modelo es aplicar dichas políticas con base en los permisos, en vez de las acciones que el usuario puede llevar a cabo. Los permisos son determinados de acuerdo al rol y a la vista correspondiente del usuario, e.g., el rol "médico" puede acceder a la vista "registro médico del paciente"; de acuerdo a las políticas de seguridad del hospital, el sistema enviará al usuario que juegue este rol toda la información acerca del paciente, mientras que en el caso del rol "enfermera", el sistema le transmitirá determinada información dado que los permisos de este rol son restringidos.

El modelo W-RBAC está enfocado en el ámbito de la administración de flujos de trabajo, donde los permisos son asignados en función de las actividades y del rol del usuario. De este modelo se desprenden dos sub-modelos, W0-RBAC y W1-RBAC. El objetivo principal del sub-modelo W0-RBAC es definir políticas que prohíban a un usuario realizar dos o más actividades en un mismo proceso, e.g., suponga el proceso "autorización de estancia de un alumno en el Departamento de Computación"; en este proceso, el rol "secretaria" tiene el permiso de registrar al alumno en el sistema, sin embargo no tiene el permiso de autorizar la estancia del alumno. El sub-modelo W1-RBAC incorpora excepciones en las políticas de seguridad.

El concepto más importante en este sub-modelo es el grado de importancia que tiene cada una de las políticas. El grado de importancia permite determinar qué políticas pueden ser eliminadas. Sin embargo, los modelos Or-BAC y W-RBAC se encargan de definir reglas para determinar las políticas de seguridad de la organización. Particularmente, el modelo Or-BAC (al igual que sistema W-RBACSoft propuesto en este trabajo de investigación) está orientado a entornos organizacionales, pero Or-BAC define los permisos solo en base al rol y a la vista del usuario, omitiendo la importancia de cada una de las sub-organizaciones, i.e., define el rol general del usuario y con ello sus permisos dentro de toda la organización. Por el contrario, el sistema W-RBACSoft se enfoca en proponer una administraión basada en áreas autónomas. Su objetivo principal es separar permisos con base en el área donde el usuario se encuentre i.e., cada área puede asignar un rol distinto. Otro de los objetivos de W-RBACSoft es definir políticas con base en el área y en la tarea a realizar. El control de acceso que ofrece W-RBACSoft pretende facilitar la administración de la organización, así como también contemplar el enfoque del modelo W0-RBAC, que se basa en la asignación de permisos en función de las actividades.

Con base en el concepto de los sistemas de flujos de trabajo, se analizó el *framework*

uFlow que permite crear un entorno ubicuo capaz de proporcionar servicios con base en la conciencia de contexto. uFlow está basado en un modelo de contexto estructural y en un lenguaje de descripción de flujos de trabajo ubicuos denominado uWDL. Este lenguaje se encarga interpretar el contexto del usuario para así ofrecerle los servicios correspondientes a su situación. Sin embargo, no implementa políticas de seguridad.

Los sistemas ABC y mobileSJ se asemejan en el apoyo que pretenden proporcionar a los trabajadores en la realización de sus actividades. La diferencia entre estos sistemas y W-RBACSoft es que este último ofrece el flujo de trabajo a realizar con base en el objetivo del usuario. ABC y mobileSJ se enfocan en entornos hospitalarios. Estos sistemas ofrecen distintas propuestas de solución para la administración de información, actividades y recursos de este entorno.

ABC es un *framework* que propone una implementación del cómputo basado en actividades, el cual permite la organización de los datos, de los servicios y de las aplicaciones a las que el usuario puede acceder. ABC proporciona información dependiente del contexto basada en "actividades", las cuales pueden ser creadas, iniciadas y reanudadas por un colaborador en cualquier dispositivo, e.g., *display* y *laptop* que se encuentre dentro del hospital. Las actividades que se llevan a cabo cuentan con su respectiva aplicación, lo cual implica que los médicos acceden a varias aplicaciones que muestran información apropiada. Sin embargo, este *framework* no está enfocado a dispositivos móviles, e.g., PDA y smartphones, debido a sus capacidades limitadas de procesamiento y despliegue.

El sistema mobileSJ propone un enfoque de cómputo basado en contexto, cuyo objetivo es satisfacer los requerimientos de los usuarios. Uno de los principales elementos que compone el sistema mobileSJ es un mecanismo que permite la localización de: 1) los trabajadores dentro del hospital, 2) los artefactos más usados y 3) los colegas que pueden colaborar en una actividad. La tecnología utilizada para determinar la actividad que deberá realizar un usuario, está basada en el entrenamiento de una red neuronal que asocia información contextual. Los dispositivos móviles representan a los colaboradores, ya que se asume que un usuario siempre está junto a su PDA, el cual le proporciona la información correspondiente a la actividad que debe llevar a cabo con base en su contexto.

Mediante el análisis de estos dos sistemas se puede decir que su objetivo es apoyar las actividades del trabajo diario de las personas. Sin embargo, ambos sistemas otorgan un rol al usuario pero no se basan en el modelo estándar RBAC ni tampoco en el uso de la tecnología de flujos de trabajo para determinar la lista de actividades de cada usuario.

La Tabla 4.1 determina la tecnología que utiliza cada sistema analizado en función de los aspectos básicos del sistema W-RBACSoft propuesto. Es claro que las áreas que contempla nuestro tema de investigación han sido atacadas en otros sistemas pero de manera separada, mientras que nuestro sistema intenta incorporarlas y adaptarlas para ofrecer un sistema de gestión de flujos de trabajo en entornos ubicuos, los cuales están estructurados en áreas autónomas.

	uFlow	ABC	mobileSJ	W-RBACSoft
Flujo de trabajo	Servicios	Actividades	Actividades	Actividades
Localización	✓	✓	✓	✓
RBAC	×	×	×	✓
Áreas autónomas	×	×	×	✓
Ubicuidad	✓	✓	✓	✓

Tabla 2.1: W-RBACSoft y los sistemas analizados

Capítulo 3

Análisis y diseño del sistema W-RBACSoft

El objetivo del presente capítulo es describir un modelado orientado a objetos del sistema W-RBACSoft (*Workflow Software - Role Based Access Control*). Este sistema pretende administrar flujos de trabajo en un entorno distribuido para apoyar a los usuarios durante su desplazamiento por las áreas de una organización. En primer lugar, se describe los principios de la administración basada en áreas de una organización, así como la propuesta de estructuración jerárquica de dichas áreas (cf. sección 3.1). Posteriormente, se presenta un escenario que pone en evidencia los beneficios que el sistema W-RBACSoft ofrece a los miembros y visitantes de una organización (cf. sección 3.2). Enseguida, se describe el análisis de dicho sistema mediante casos de uso, así como su arquitectura de software utilizando un diagrama de bloques (cf. sección 3.3). A continuación, se presenta el diseño del sistema W-RBACSoft por medio de diagramas de clases (cf. sección 3.4). Finalmente, se describe sus arquitecturas de distribución y de comunicación empleando diagramas de secuencia (cf. sección 3.5).

3.1 Administración basada en áreas de una organización

Hoy en día las grandes organizaciones están divididas en sub-organizaciones con el objetivo de ser más eficientes y de proporcionar a los usuarios servicios más especializados [Roth, 2006]. Una sub-organización está definida como una área autónoma. De esta manera, cada área se convierte en una unidad de administración encargada del manejo de sus propios recursos (e.g., servicios, flujos de trabajo y roles) con base en las políticas de seguridad de la organización. La propuesta de una administración basada en áreas facilita la gestión de la organización, ya que sus recursos son manejados y controlados de manera distribuida.

Las distintas áreas que componen la organización intercambian información, con el fin de ayudar a los usuarios nómadas a lograr sus objetivos parciales y globales. Desde el punto de vista físico, un área autónoma puede corresponder a un edificio, a varios edificios o a una parte de un edificio.

La estructuración de las áreas que conforman una organización está basada en un modelo jerárquico, donde los niveles altos proporcionan flujos de trabajo con actividades generales, mientras que los niveles bajos los ofrecen de una manera más específica. La Figura 3.1 ilustra la asignación de un rol dentro de una jerarquía de áreas, donde A es un área de alto nivel y B es un área de bajo nivel.

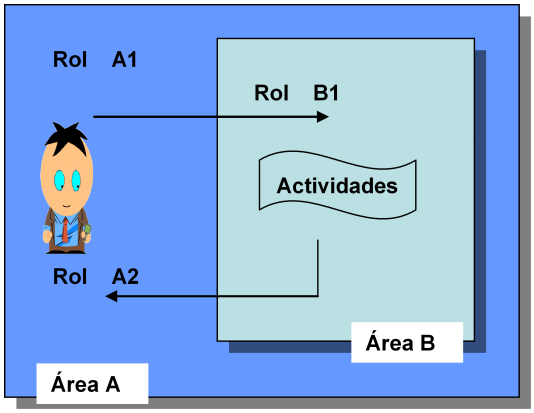

Figura 3.1: Roles asignados a un usuario en áreas autónomas jerárquicas

Cada área define un conjunto de roles, e.g., el área A define los roles A1 y A2, mientras que el área B define el rol B1. Cuando el usuario nómada accede al área A, esta le asigna el rol A1, en función del cual el usuario realiza un conjunto de actividades. Posteriormente, el usuario se dirige al área B, la cual le atribuye el rol B1 así como las actividades que debe llevar a cabo. Una vez que el usuario termina estas actividades, regresa al área A, la cual le asigna el rol A2, i.e., el usuario puede obtener un rol distinto al que obtuvo cuando entró por primera vez al área A.

En función de la estructuración de las áreas, se propone una distribución jerárquica que permite una mejor administración, así como un mejor control de los servicios, recursos y actividades de la organización. El modelo de estructuración de las áreas imita al esquema de un árbol (conjunto de nodos conectados), donde las áreas están representadas por nodos, los cuales a su vez pueden estar formados por cero o más nodos. Se dice que el área A es padre del área B si existe un enlace de A a B (ver Figura 3.2). Solo puede haber un único nodo sin padres, el cual es llamado nodo raíz, que corresponde al área principal de la organización.

Por otra parte, las grandes organizaciones requieren un sistema de flujo de trabajo sofisticado para administrar sus tareas [Han et al., 2006]. La administración de un sistema de flujo de trabajo global es muy compleja debido a las particularidades de las políticas de cada sub-organización. Por lo tanto, entre otras consideraciones, resulta muy interesante tomar los beneficios de la administración basada en áreas de una organización, para definir y administrar un sistema de flujo de trabajo de forma distribuida. De esta manera, las áreas ascendentes proporcionan flujos de trabajo generales, mientras que las áreas descendentes proporcionan flujos de trabajo más específicos.

Es importante que cada área se encargue de la administración y seguridad de su información para evitar anomalías. Sin embargo, para los usuarios ajenos a la organización puede resultar complicada la realización de tareas que engloben varias áreas, ya que des-

conocen las actividades que pueden ser realizadas dentro de ellas. Esta limitación puede provocar que los objetivos del usuario no se logren exitosamente.

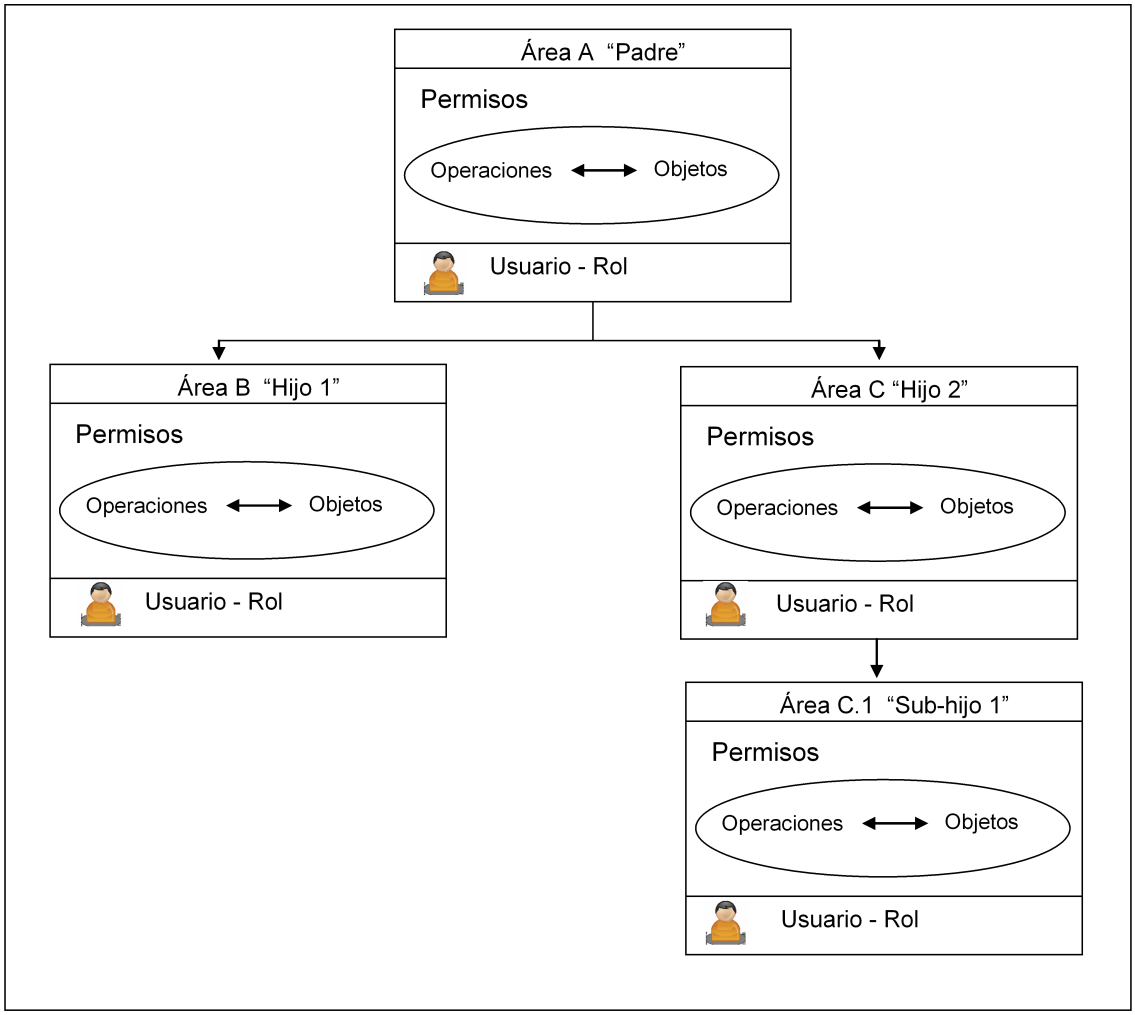

Figura 3.2: Módelo jerárquico de áreas autónomas

Para tratar de hacer frente al problema mencionado anteriormente, se propone el sistema W-RBACSoft (*Workflow Software - Role Base Access Control*), el cual ofrece una administración basada en áreas de una organización y una gestión distribuida de flujos de trabajo. Mediante el siguiente escenario se pretende poner en evidencia los beneficios que este sistema ofrece a los miembros y visitantes de una organización.

3.2 Escenario de uso del sistema W-RBACSoft

Suponga una institución educativa, la cual está constituida por varias áreas que corresponden a una profesión distinta. El Dr. Medina ha sido invitado a dar una conferencia en el `Departamento de Computación` a las 10:00 A.M. y a impartir un curso en el `Departamento de Mecatrónica` a las 12:00 P.M. El sistema W-RBACSoft se encarga de

administrar el flujo de trabajo que debe llevar a cabo el Dr. Medina, así como también de determinar las áreas a las que debe acceder (ver Figura 3.3).

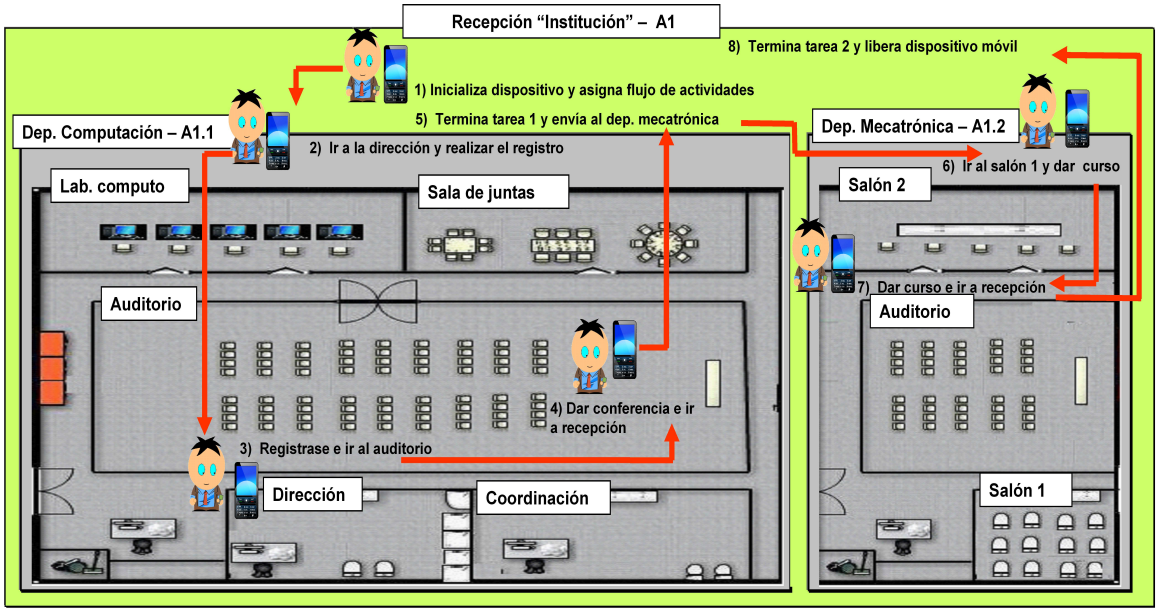

Figura 3.3: Escenario de uso del sistema W-RBACSoft en una institución educativa

Cuando el Dr. Medina llega al área principal de la institución, se autentifica (i.e., proporciona su nombre, procedencia, perfil y objetivo) y el sistema W-RBACSoft lo registra en su base de datos. En función de la información recibida del Dr. Medina, el sistema se encarga de asignarle el identificador `id1` y el rol `investigadorInvitado`, así como de establecer el permiso `utilizar` un `dispositivo móvil`.

Una vez registrado el Dr. Medina, se le proporciona un dispositivo móvil donde recibirá la información correspondiente a las dos tareas que debe realizar dentro de la institución: 1) `dar` una `conferencia` en el `Departamento de Computación` e 2) `impartir` un `curso` en el `Departamento de Mecatrónica`.

El área de `Recepción` envía, al dispositivo móvil del Dr. Medina, el flujo de trabajo general que define las actividades siguientes: 1) `acceder` al `Departamento de Computación`, 2) `acceder` al `Departamento de Mecatrónica`, 3) `regresar` a la `Recepción` y 4) `entregar` el `dispositivo móvil`. El sistema W-RBACSoft informa al `Departamento de Computación` y al `Departamento de Mecatrónica` que el Dr. Medina accederá a ellos. El flujo de trabajo que proporciona la `Recepción` es general, ya que solo específica las dos áreas a las que podrá acceder el Dr. Medina para realizar sus tareas. Cuando el usuario acceda a cada una de las áreas, estas ya tendrán conocimiento de su llegada y le proporcionarán un nuevo flujo de trabajo con actividades más específicas.

Cuando el Dr. Medina llega al `Departamento de Computación`, esta área le atribuye el rol `investigadorInvitado-al-Departamento-de-Computación`, los permisos `acceder` al `Auditorio` y `utilizar` un `proyector` y el flujo de trabajo 1) `ir` a la `Dirección`, 2) `registrarse`, 3) `ir` al `Auditorio` y 4) `dar` una `conferencia`. Una vez finalizada esta tarea, el Dr. Medina procede a dirigirse al `Departamento de Mecatrónica` para realizar la tarea de `impartir` un `curso`. Esta área le asigna el rol `investigadorInvitado-al-De-`

`partamento-de-Mecatrónica`, los permisos `acceder` al `Salón 1` y `utilizar` un `proyector` y el flujo de trabajo 1) `ir` al `Salón 1` e 2) `impartir` un `curso`. Al terminar las actividades de este flujo de trabajo, el Dr. Medina se dirige a la `Recepción` para devolver el dispositivo móvil y finalmente salir de la organización. De esta manera, el sistema W-RBACSoft da por concluído el proceso de este visitante.

3.3 Análisis del sistema W-RBACSoft

Para poder administrar los sub-flujos de trabajo que conforman el flujo de trabajo general de una organización, se requiere que las áreas encargadas de administrar dichos sub-flujos de trabajo conozcan los estados del flujo de trabajo general. De esta manera, las áreas involucradas solo estarán en espera de la llegada del usuario nómada para generarle un flujo de trabajo más específico. El sistema W-RBACSoft se encarga de notificar el objetivo del usuario a cada una de las áreas involucradas por medio del servidor W-RBACSoft asociado a cada una de ellas. En esta sección, se describe los principales casos de uso del sistema W-RBACSoft, así como su arquitectura de software.

3.3.1 Casos de uso

Con el fin de ilustrar de manera más detallada las funciones que realiza cada uno de los actores que interactúa con el sistema W-RBACSoft, se presenta los siguientes casos de uso.

1. El actor `Administrador` es una persona que se encarga de validar el acceso del usuario a la organización. Las funciones que realiza el `Administrador` son ajenas al sistema W-RBACSoft, ya que se encarga de solicitar la identificación del usuario para determinar si le autoriza o niega el acceso. El `Administrador` realiza las siguientes funciones (ver Figura 3.4):

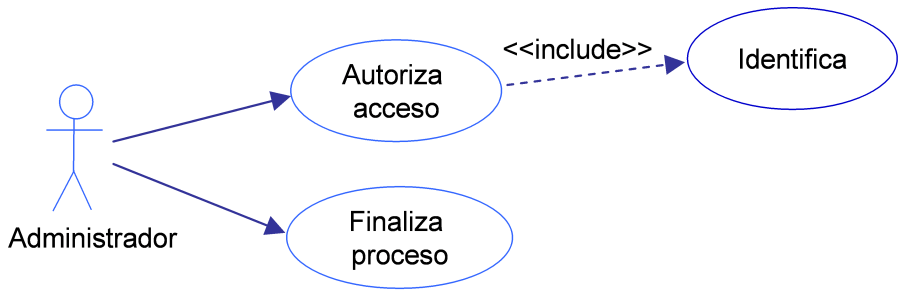

Figura 3.4: Casos de uso del actor `Administrador`

   (a) **Autoriza acceso:** esta función se lleva a cabo siempre y cuando se realice la sub-función **Identifica**, la cual verifica si la información que proporciona el usuario es correcta o no. Si la información es correcta, la función **Autoriza acceso** permite el acceso del usuario a la organización.

(b) **Finaliza proceso:** está función se lleva a cabo cuando: a) al usuario se le niega el acceso, b) el usuario realiza alguna anomalía dentro de la organización o c) el usuario concluye sus actividades.

2. La persona que representa al actor `Usuario` interactúa primeramente con el `Administrador` y después con el servidor W-RBACSoft del área correspondiente. El `Usuario` realiza las siguientes funciones (ver Figura 3.5):

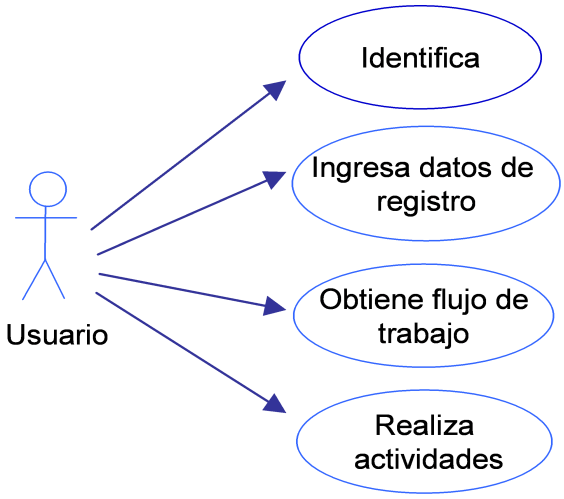

Figura 3.5: Casos de uso del actor `Usuario`

(a) **Identifica:** el `Usuario` proporciona su identificación.

(b) **Ingresa datos de registro:** el `Usuario` ingresa su información correspondiente tal como nombre, procedencia, perfil y objetivo.

(c) **Obtiene flujo de trabajo:** cuando el servidor del sistema W-RBACSoft recibe la información necesaria del `Usuario`, envía el flujo de trabajo al dispositivo móvil que el `Administrador` le proporcionó.

(d) **Realiza actividades:** el `Usuario` lleva a cabo cada una de las actividades del flujo de trabajo que el sistema asignó.

3. Cuando el `Usuario` llega al área, que corresponde a la entrada de la organización, interactúa con el `Administrador` para solicitar acceso. Una vez autorizado el acceso, el `Usuario` interactúa directamente con el servidor W-RBACSoft (ver Figura 3.6).

(a) **Usuario - Administrador**

- El `Usuario` interactúa con el `Administrador` principalmente en la función **Identifica** ya que, con base en la información que proporciona el `Usuario`, el `Administrador` puede autorizar o denegar el acceso. En el primer caso (autorización), el `Administrador` realiza la función **Autoriza acceso**. En el segundo caso (denegación), el `Administrador` lleva a cabo la función **Finaliza proceso**.

- Una vez que el `Administrador` realizó la función **Autoriza acceso**, el `Usuario` puede llevar a cabo la función **Ingresa datos de registro**.

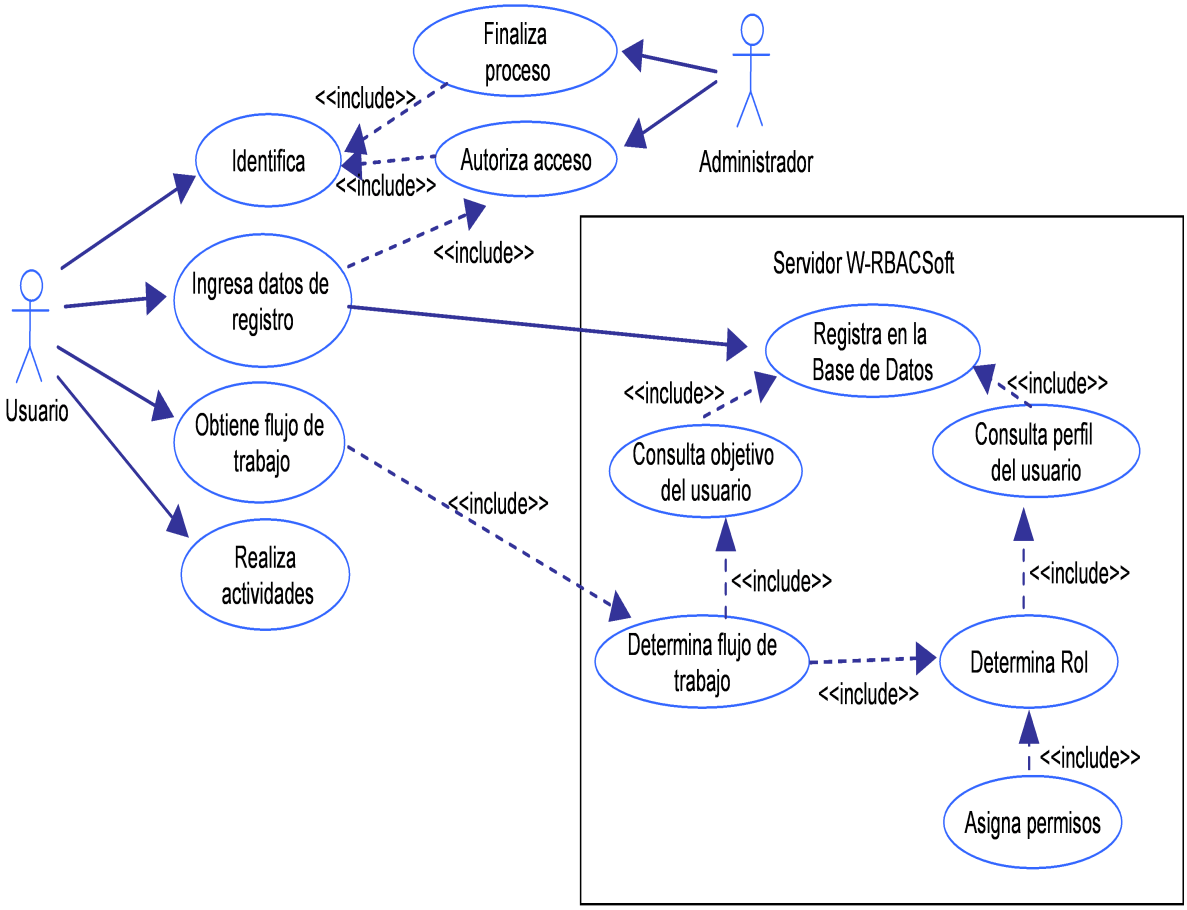

Figura 3.6: Interacción entre los casos de uso `Usuario` - `Administrador` y `Usuario` - servidor W-RBACSoft

(b) **Usuario - Servidor W-RBACSoft**

- El `Usuario` proporciona sus datos (nombre, procedencia/función, perfil y objetivo) cuando realiza la función **Ingresa datos de registro**. Estos datos son enviados al servidor W-RBACSoft del área correspondiente. Cuando este último obtiene los datos del `Usuario` ejecuta la operación **Registra en la Base de Datos** para almacenarlos de manera persistente.
- Una vez que el servidor W-RBACSoft registró los datos del `Usuario`, procede a llevar a cabo las funciones necesarias para definir las actividades que el `Usuario` debe realizar. Estas funciones son **Consulta perfil del usuario** y **Consulta objetivo del usuario**, las cuales desglosan las subfunciones siguientes:
  i. **Determina rol:** el servidor W-RBACSoft debe realizar esta función para asignar al `usuario` los permisos y las restricciones que tendrá

dentro de la organización. El rol se define de acuerdo al perfil del `Usuario`, el cual es obtenido mediante una consulta a la base de datos que contiene información de dicho `usuario`; esta consulta se lleva a cabo mediante la sub-función **Consulta perfil del usuario**.

ii. **Determina flujo de trabajo:** para efectuar esta función es necesario realizar la sub-función **Consulta objetivo del usuario**, la cual hace una consulta a la base de datos para obtener esta información. Una vez que el servidor W-RBACSoft obtuvo el valor de la variable contextual `objetivo`, define las actividades que debe y puede realizar el `Usuario`.

Cabe mencionar que una vez que el `Usuario` tiene asignado un rol y un flujo de trabajo, ejecuta la función **Realiza actividades**.

3.3.2 Arquitectura de software del sistema W-RBACSoft

El sistema W-RBACSotf está compuesto por una aplicación cliente y un servidor, los cuales intercambian información cuando así se requiere. Cabe mencionar que cada área que conforma la organización cuenta con su propio servidor W-RBACSoft. El software del servidor W-RBACSoft asociado a un área es semejante al de las demás áreas, la única diferencia es que cada uno maneja su propia información y recursos.

La Figura 3.7 ilustra la arquitectura del sistema W-RBACSoft y los componentes que la conforman, así como la organización de las áreas. La aplicación cliente W-RBACSoft se encarga de la interacción con el usuario, ya que este componente representa el proceso que inicia el usuario cuando accede al sistema W-RBACSoft. Como ya se mencionó, cada área tiene su propio servidor W-RBACSoft, el cual se comunica con los servidores de las demás áreas para mantener el control de las actividades del usuario.

A su vez, el servidor W-RBACSoft está formado por: a) el componente `RBAC`, el cual está encargado de controlar el acceso al área y de determinar el rol que asignará al usuario así como sus permisos y restricciones, b) el componente `Workflow`, que define el flujo de trabajo del usuario y c) el componente `Comunicación`, el cual se encarga de establecer la comunicación con las áreas involucradas en el flujo de trabajo correspondiente. El servidor W-RBACSoft asociado a un área contiene una base de datos para almacenar la información correspondiente de dicha área, así como el registro de cada usuario que tenga acceso a ella.

Con la finalidad de ilustrar la arquitectura del software del sistema W-RBACSoft, consideré el siguiente ejemplo. Cuando el usuario llega al área padre, presenta una identificación al administrador del área el cual inicializa la aplicación cliente W-RBACSoft. El usuario se registra mediante esta aplicación, la cual envía los datos de autentificación al servidor W-RBACSoft del área correspondiente (ver Figura 3.7 ref. # 1). En respuesta, el servidor almacena los datos del usuario (e.g., nombre, perfil, objetivo) en su base de datos (ver Figura 3.7 ref. # 2) y envía la variable contextual `estatus` al componente RBAC (ver Figura 3.7 ref. # 3 y # 4). Este componente asigna el rol al usuario y determina los permisos correspondientes mediante una consulta a la base de datos (ver Figura 3.7 ref. # 5). Posteriormente, el componente `Workflow` determina el flujo de trabajo de acuerdo al valor de la variable contextual `objetivo`, en tanto que el componente `Comunicación` informa a las áreas involucradas en el flujo de trabajo del usuario para que estas tengan

conocimiento de la llegada del usuario y generen el flujo de trabajo correspondiente (ver Figura 3.7 ref. # 6 y # 7). Finalmente el servidor W-RBACSoft envía el flujo de trabajo al dispositivo móvil del usuario nómada (ver Figura 3.7 ref. # 8).

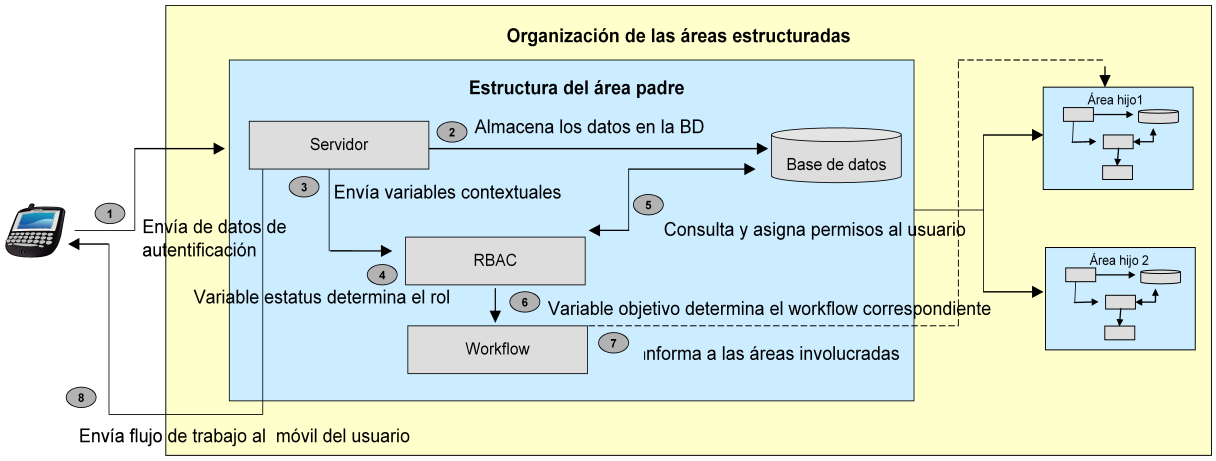

Figura 3.7: Arquitectura de software del sistema W-RBACSoft

3.4 Diseño del sistema W-RBACSoft

Un área autónoma es una entidad que proporciona servicios especializados para administrar sus propios recursos de una manera independiente. Para definir la estructura de cada área, se determinaron las siguientes clases: `Jerarquía_de_ áreas`, `Usuario`, `Acceso`, `Workflow` y `Comunicación`. Estas clases realizan las siguientes funciones:

- Clase `Jerarquía_de_áreas`: establece una relación entre el área actual y el área que asciende o desciende de ella. Con base en la estructura jerárquica de las áreas, este componente proporciona el siguiente conjunto de operaciones: crear sub-areas (descendientes), unir sub-áreas y eliminar sub-áreas.

- Clase `Usuario`: se encarga de guardar los datos correspondientes al usuario, así como la información contextual del área, tal como: rol, flujo de trabajo y áreas a las que accederá.

- Clase `Acceso`: es responsable determinar los permisos del usuario nómada con base en el rol asignado. Sus principales operaciones son: asignar un rol a los usuarios nómadas, modificar, activar y desactivar el rol asignado, administrar el conjunto de permisos de cada rol, asignar el conjunto de permisos de cada recurso y establecer las obligaciones del usuario dentro del área.

- Clase `Workflow`: determina un conjunto de flujos de trabajo, cada vez que un usuario acceda a una área, identifica su objetivo y con base a él se obtiene el flujo de trabajo correcto.

- Clase `Comunicación`: se encarga de establecer la conexión entre las áreas involucradas.

La Figura 3.8 muestra las relaciones entre las clases previamente mencionadas. Estas relaciones se explicarán de manera más detallada en la descripción de cada una de las clases principales del sistema W-RBACSoft (cf. secciones 3.4.1-3.4.5). La clase `Área` posee una relación con las clases: a) `Acceso`, b) `Workflow`, c) `Comunicación` y d) `Área`, y a su vez la clase `Usuario` hace referencia a ella. Así mismo, la clase `Acceso` requiere de las clases `Rol`, `Permiso`, `Recurso` y `Operación`, ya que el usuario al acceder a un área requiere un rol y permisos, los cuales definen las operaciones que puede realizar sobre los recursos del área. Como se puede apreciar en la Figura 3.8, cada una de las clases posee referencias hacia las otras clases, dependiendo de las relaciones que necesite cada una para llevar a cabo su función.

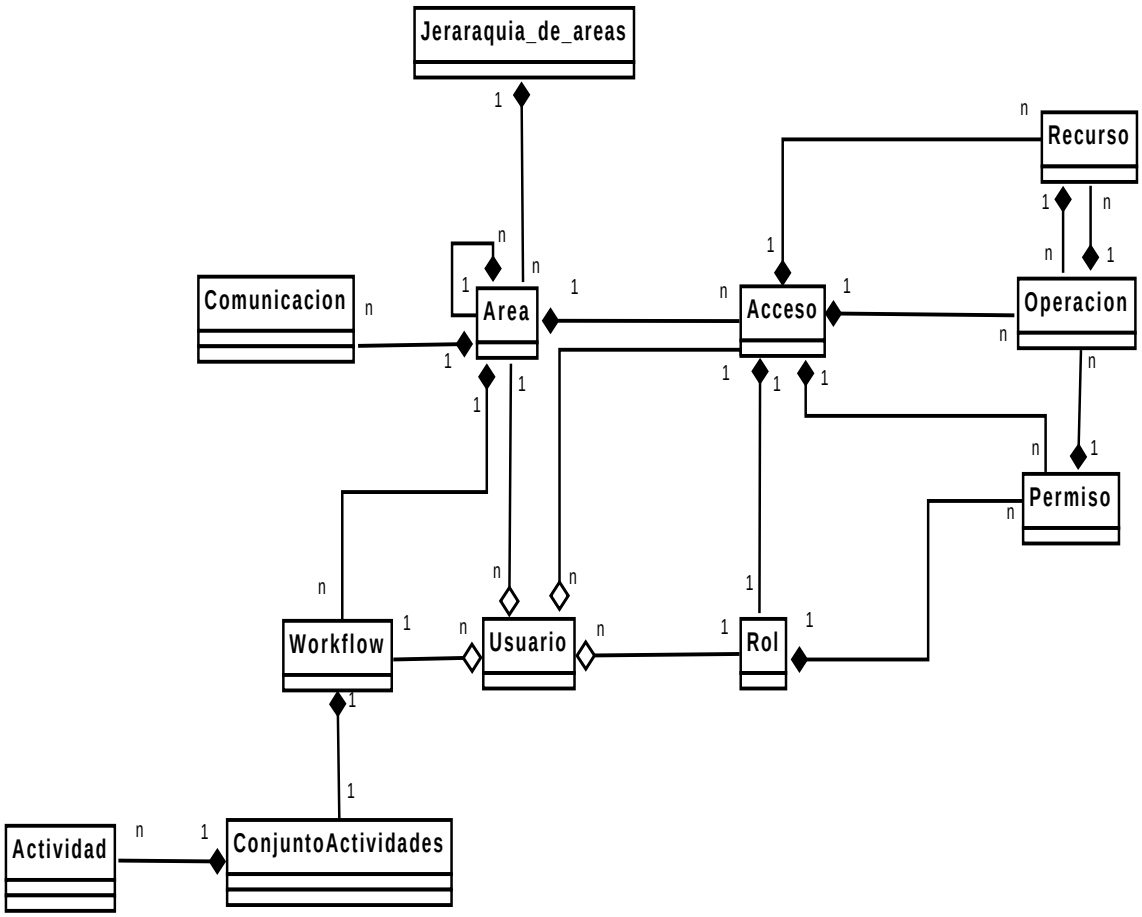

Figura 3.8: Diagrama de clases que componen el sistema W-RBACSoft

3.4.1 Clase `Jerarquía_de_áreas`

Esta clase se encarga del control de las áreas y de su organización jerárquica. La Figura 3.9 ilustra la clase `Jerarquía_de_áreas` y su relación con la clase `Área`. La clase `Jerarquía_de_áreas` se encarga de determinar el nodo raíz y de crear áreas descendentes; por esta

razón, requiere la clase **Área**, la cual se encarga de obtener los roles, los recursos y las operaciones que contempla el área, así como el rol correspondiente del usuario creando una instancia de la clase `Acceso`. De esta manera se obtiene los permisos y las áreas de acceso del usuario.

Las clases **Área** y `Jerarquía_de_áreas` mantienen una relación de composición con cardinalidad `n...1`, debido que la clase **Área** representa los nodos con los que cuenta la clase `Jerarquía_de_áreas`. Por esta razón, se dice que un área solo puede tener un área padre. Por otro lado, un área padre puede tener varias áreas hijo.

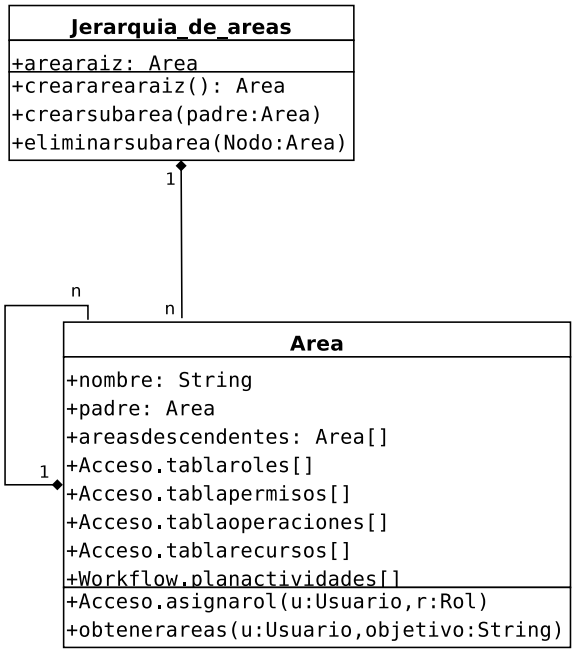

Figura 3.9: Clase `Jerarquía_de_áreas` asociada a la clase **Área**

La clase `Jerarquía_de_áreas` está formada por el atributo `arearaiz`, el cual se encarga de almacenar un nodo definido como raíz que puede ser un área o subárea. El método `crearaiz()`, como su nombre lo indica, crea el nodo raíz (nodo padre) de la jerarquía; por otro lado el método `crearsubarea()` crea una subárea a partir de un nodo raíz o de un subárea y finalmente el método `eliminarsubarea()` elimina la subárea especificada.

La clase **Área** determina el nodo padre, así como los nodos descendientes de este. El atributo `nombre` se refiere al área actual; el atributo `padre` es el nodo padre del área actual; el atributo `areasdescendientes` determina las áreas descendientes a las que se denomina áreas hijo; los atributos `Acceso.tablaroles[]`, `Acceso.tablaoperaciones[]`, `Acceso.tablaobjetos[]` almacenan respectivamente los roles, las operaciones y los recursos disponibles en el área, mientras que los atributos `Acceso.tablapermisos[]` y `Workflow.planactividades[]` almacenan los permisos y las actividades asignadas al usuario. El método `Acceso.asignarol()` asigna un rol al usuario nómada; mientras que el método `obtenerareas()` determina las áreas que accederá el usuario con base en su objetivo. La clase **Área** invoca métodos de las clases `Acceso` y `Workflow`.

Una estructuración jerárquica de las áreas permite definir y administrar, de manera eficiente, los recursos y los flujos de trabajo de una organización. De esta manera, una

organización contaría con un flujo de trabajo que estaría lógicamente distribuido entre sus sub-áreas las cuales lo gestionarían de manera autónoma. Suponga una estructuración jerárquica en la que cada área está encargada de gestionar sus propios componentes del flujo de trabajo. Cada componente del flujo de trabajo de un área define sus puntos de conexión, i.e., los puntos de entrada y salida del sub-flujo de trabajo. Por ejemplo, la parte central de la Figura 3.10 presenta los componentes del flujo de trabajo de las áreas A1 (nodo raíz), A3 y A.3.1. En la parte derecha se define de manera más detallada el flujo de trabajo que genera cada una de las áreas con base en un determinado objetivo. De esta manera, cada área constituye una parte independiente del flujo de trabajo global. Los puntos de conexión del área A3 son C1 y C2, lo cuales respectivamente están conectados con el área A1 mediante los puntos de conexión I1 y I2.

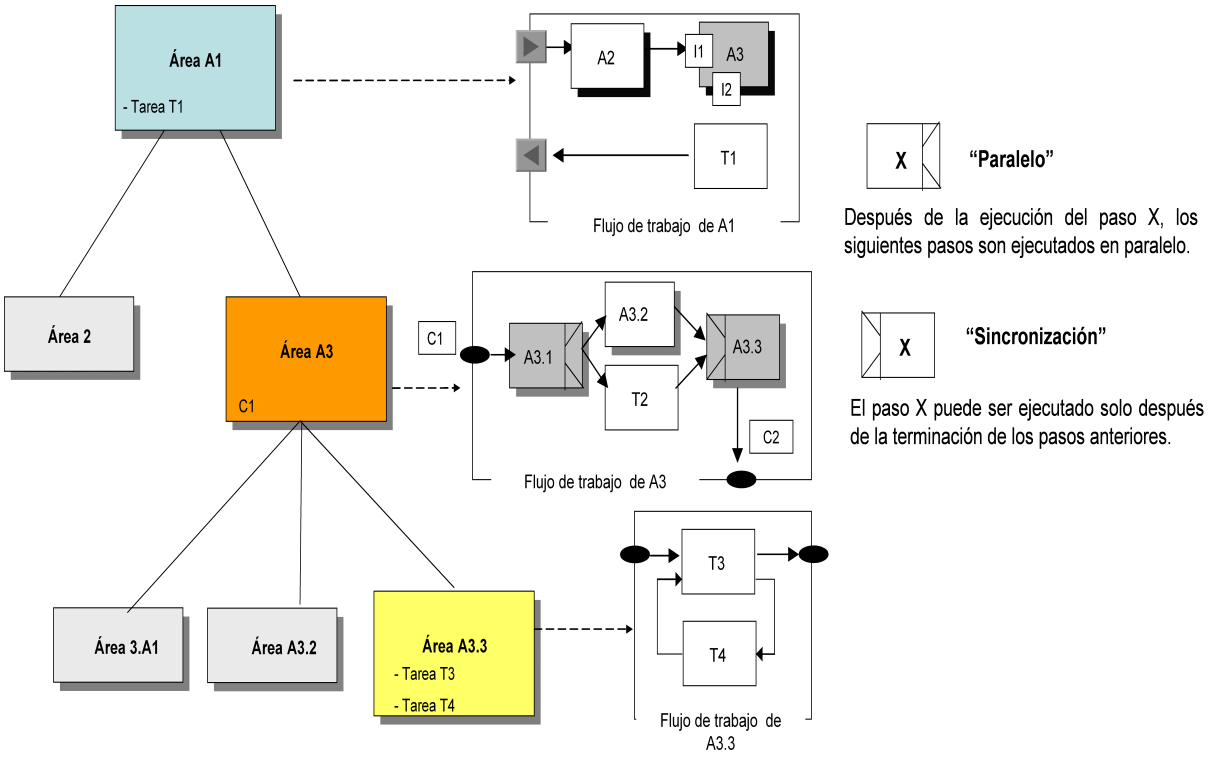

Figura 3.10: Estructuración de áreas autónomas

De esta manera, se define y administra eficientemente el flujo de trabajo global, debido a que los sub-flujos de trabajo se administran de forma autónoma. La Figura 3.10 presenta una jerarquía de áreas autónomas, donde el área A1 incluye dos subáreas A2 y A3, mientras que el área A3 engloba tres subáreas A3.1, A3.2 y A3.3. Cada área administra su información contextual, recursos, roles y los componentes de sus sub-flujos de trabajo.

El área A1 define un flujo de trabajo secuencial que ejecuta sucesivamente los flujos de trabajo de las áreas A2 y A3, al igual que la tarea T1. El área A3 contiene un sub-flujo de trabajo que consiste en la ejecución del sub-flujo de trabajo del área A3.1, donde posteriormente se ejecutan en paralelo: a) el sub-flujo de trabajo del área A3.2 y b) la tarea T2 dentro del área A3. El área A3 termina con un flujo de trabajo secuencial y define el punto de inicio de ejecución del sub-flujo de trabajo del área A3.3. El ejemplo

es ilustrado en la Figura 3.10 destaca el principio de la administración independiente y distribuida de áreas, que muestra la eficacia para administrar un flujo de trabajo global.

3.4.2 Clase `Usuario`

La clase `Usuario` se encarga de almacenar la información correspondiente del usuario tal como: su identificador, su perfil, su objetivo, su rol y su flujo de trabajo. Esta información es obtenida cuando el usuario accede al área. Este tipo de información se obtiene mediante la instancia de las clases: a) `Área`, b) `Acceso`, c) `Rol` y d) `Workflow` (ver Figura 3.11).

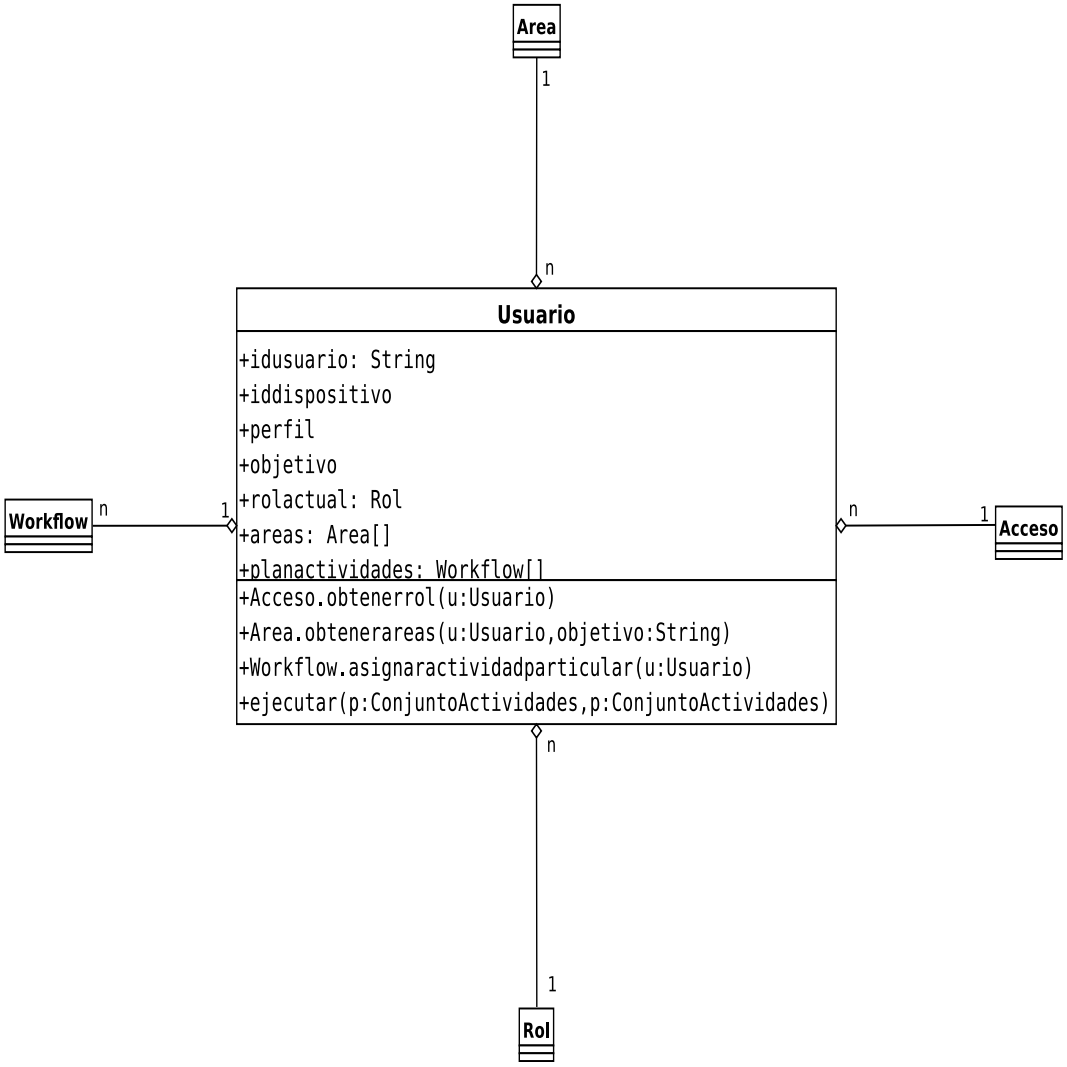

Figura 3.11: Clase `Usuario`

Las clases `Usuario` y `Área` mantienen una relación de agregación con cardinalidad `n...1`. Como se ha venido mencionando cada área es responsable del acceso de sus usuarios, por lo tanto un usuario solo tienen acceso al área actual. Por otro lado, un área puede dar acceso a varios usuarios.

Entre las clases `Usuario` y `Acceso` existe una relación de agregación con cardinalidad `n...1`, debido a que un usuario requiere de un control de acceso a un área (el sistema le asigna al usuario su rol y sus permisos). Por otra parte, un control de acceso se determina para cada usuario que entra a un área.

Las clases `Usuario` y `Rol` tienen una relación de agregación con cardinalidad `n...1`, ya que un usuario requiere de la asignación de un rol dentro de un área. Por otro lado, un rol puede ser asignado a varios usuarios.

Las clases `Usuario` y `Workflow` definen una relación de agregación con cardinalidad `n...1`, porque un usuario requiere un flujo de trabajo (*Workflow*) dentro de un área. Por otra parte, un flujo de trabajo puede ser asignado a varios usuarios.

Los atributos y métodos correspondientes a está área son los siguientes:

1. Atributos de la clase `Usuario`:

   - `idusuario`: guarda el identificador del usuario;
   - `iddispositivo`: almacena el identificador del dispositivo proporcionado en el área principal;
   - `perfil`: corresponde al perfil del usuario;
   - `rolactual`: corresponde al rol actual dentro del área actual;
   - `areas`: es un arreglo que se encarga de almacenar las áreas que accederá el usuario de acuerdo a su objetivo; y
   - `planactividades`: corresponde a la lista de actividades a realizar dentro de un área.

2. Métodos de la clase `Usuario`:

   - `Acceso.obtenerrol()`: crea una instancia de la clase `Acceso` para determinar su rol correspondiente;
   - `Área.obtenerareas()`: obtiene las áreas a las que accedará el usuario, instanciando la clase `Área`;
   - `Workflow.asignaractividadparticular()`: obtiene las actividades que el usuario llevará a cabo mediante una instancia de la clase `Workflow`; y
   - `ejecutar()`: determina qué actividad debe ser ejecutada en el estado actual del usuario.

3.4.3 Clase `Acceso`

El modelo de control de acceso utilizado en el sistema W-RBACSoft permite definir qué permisos y restricciones tiene un usuario nómada dentro de la organización, específicamente en el área actual donde se encuentra. RBAC es un modelo que tiene el potencial de: 1) reducir la complejidad y el costo de la administración de seguridad y 2) determinar si se puede conceder un tipo particular de acceso a un recurso o información, asignándole uno o más roles a cada usuario (ver Figura 3.12).

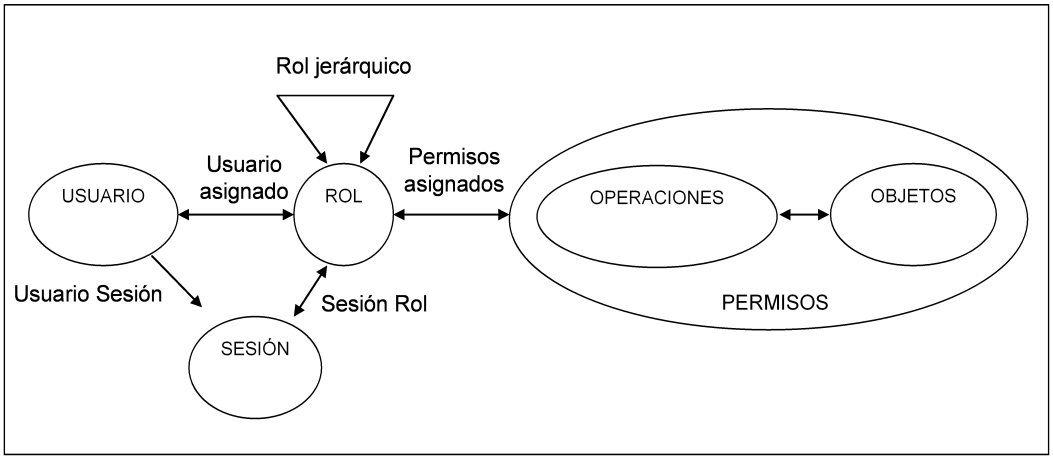

Figura 3.12: Modelo de control de acceso basado en roles (RBAC)

Los roles son definidos por cada área, los cuales corresponden con los perfiles o cargos definidos en cada una de ellas. A cada rol se le asignan funciones las cuales a su vez pueden contener otras funciones, i.e., un rol puede tener asignadas tareas y permisos que están asociados a otro rol. Cuando esto sucede, se establece una jerarquía de roles.

Modelado del RBAC

La clase `Acceso`, mostrada en la Figura 3.13, establece el modelo para la administración del acceso dentro de un área. Esta clase está basada en el modelo RBAC, ya que cuenta con las clases `Usuario`, `Roles`, `Permisos`, `Operaciones`, `Recursos` y `Jerarquía de Roles`. En el trabajo propuesto por Ravi et al. [Ravi et al., 1996] el modelo RBAC está basado en el lenguaje UML, el cual contiene varias clases para cada elemento de RBAC. Se tomaron algunas partes de este esquema y se agregaron nuevos aspectos relacionados con la administración de acceso que requiere el sistema W-RBACSoft.

De acuerdo con el modelo RBAC se determina que las clases `Rol`, `Permiso`, `Operación` y `Recurso` forman parte de la clase `Acceso`, ya que el usuario al acceder a una área se le asigna su rol, sus permisos y sus respectivas operaciones sobre los objetos. Las relaciones entre las clases `Acceso`, `Rol`, `Permiso`, `Operacion` y `Recurso` son las siguientes:

- Las clases `Acceso` y `Rol` mantienen una relación de composición con cardinalidad de `1...1`, ya que para determinar un tipo de acceso a un usuario se le debe asignar un rol. Por otro lado, un rol asignado define un solo tipo acceso.

- Entre las clases `Acceso` y `Permiso` existe una relación de composición con cardinalidad de `1...n`, porque un tipo de acceso puede atribuir varios permisos. Sin embargo, el conjunto de permisos es asignado a un tipo de acceso.

- Entre las clases `Acceso` y `Operacion` se establece una relación de composición con cardinalidad `1...n`, ya que el tipo de acceso determina un conjunto de operaciones. Por otra parte, el conjunto de operaciones se le determinan a un solo tipo de acceso.

- Las clases `Acceso` y `Recurso` definen una relación de composición con una cardinalidad `1...n`, porque un tipo de acceso determina los recursos correspondientes. Por

otro lado, estos recursos le corresponden a un tipo de acceso.

La clase más importante es la de `Acceso`, ya que da la pauta para proceder a asignar primeramente un rol a un usuario y posteriormente sus respectivos permisos. A continuación, se describe los atributos y métodos más relevantes de la clase `Acceso`, la cual crea instancias de las clases `Rol`, `Permiso`, `Operacion` y `Recurso`.

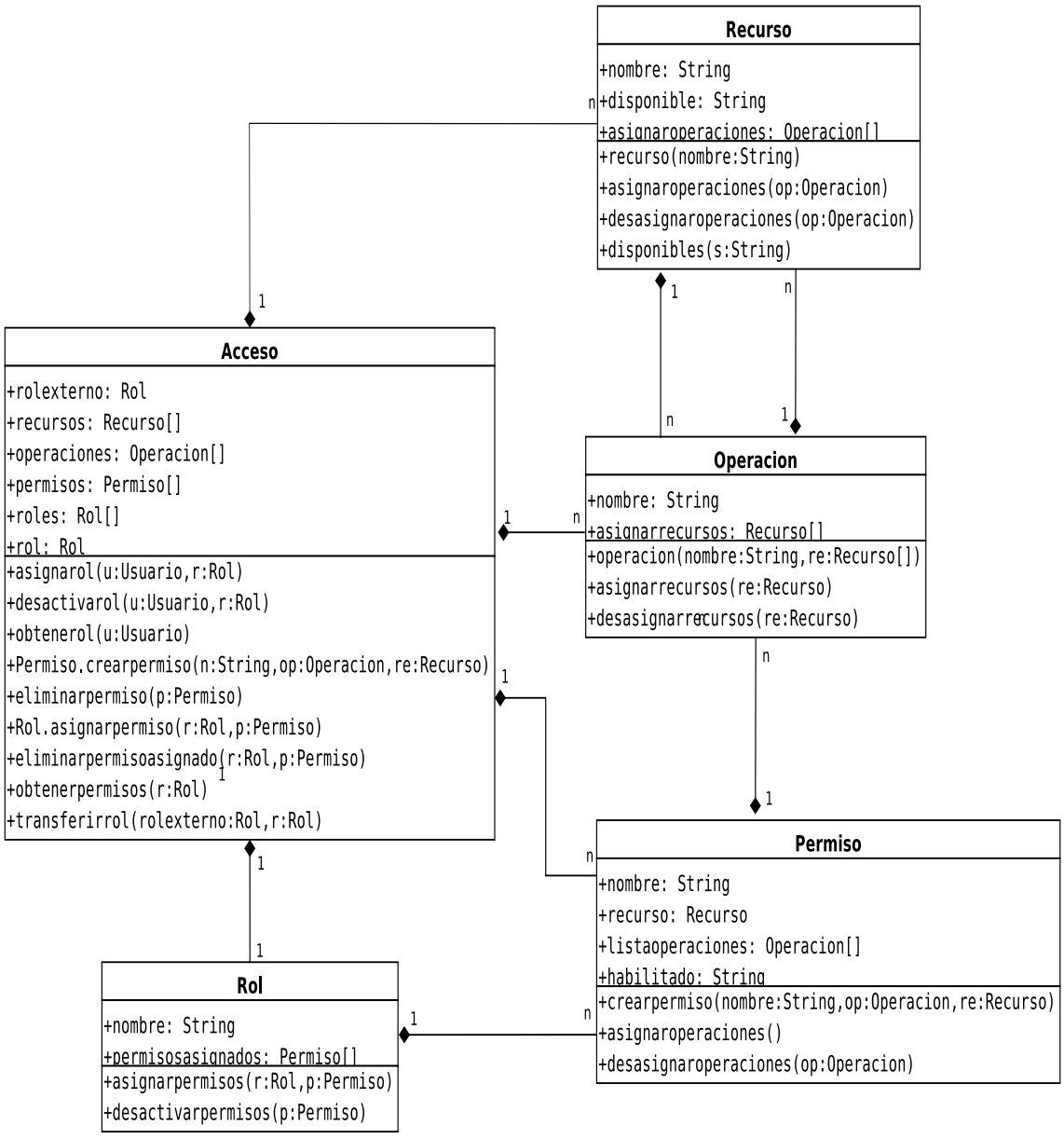

Figura 3.13: Clases `Acceso`, `Rol`, `Permiso`, `Operacion` y `Recurso`

1. Atributos de la clase `Acceso`:

   - `rolexterno`: guarda el rol que fue asignado a un usuario en un área vecina;

- `recursos`: es un arreglo que instancia la clase `Recurso` y especifica los recursos del área actual, e.g., proyector e impresora;
- `operaciones`: es un arreglo que instancia la clase `Operacion` y define las operaciones que pueden ser aplicadas a los recursos de una área, e.g., encender un proyector, imprimir un archivo;
- `permisos`: es un arreglo que instancia la clase `Permiso`, y determina los permisos de un usuario en un área. Los permisos establecen una operación que puede ser aplicada a un recurso, e.g., utilizar el proyector del laboratorio del Departamento de Computación;
- `roles`: es un arreglo que se encarga de almacenar los roles válidos dentro de una área, instanciando la clase `Rol`; y
- `rol`: almacena el rol actual de un usuario.

2. Métodos de la clase `Acceso`:

- `asignarol()`: asigna a un usuario un rol específico para establecer los permisos correspondientes a este rol;
- `desactivarol()`: desactiva el rol asignado a un usuario;
- `obtenerol`: recupera el rol de un usuario;
- `Permiso.crearpermiso()`: crea permisos (mediante el método `crearpermisos()` de la clase `Permiso`) dentro de un área; un permiso relaciona una operación con un recurso;
- `elimininarpermiso()`: elimina un permiso del arreglo `permisos`;
- `Rol.asignarpermiso()`: crea una instancia de la clase `Rol` e incluye un nuevo permiso en el arreglo `permisos` en función del rol actual que tiene un usuario;
- `eliminarpermisoasignado()`: elimina un permiso del arreglo `permisos`;
- `obtenerpermisos()`: obtiene los permisos referentes a un rol; y
- `transferirrol()`: hereda los permisos del rol anterior de un usuario al nuevo rol asignado.

3.4.4 Clase `Workflow`

Las principales funciones que lleva a cabo está clase son las siguientes:

- asigna tareas o actividades a un usuario nómada;
- automatiza la secuencia de las actividades de los flujos de trabajo de un usuario nómada;
- controla las actividades de los flujos de trabajo; y
- apoya a un usuario nómada para alcanzar sus objetivos mediante la definición de un flujo de trabajo a realizar.

Un flujo de trabajo se determina por el valor de una variable de contexto llamada `objetivo`, la cual es obtenida cuando el usuario nómada se autentifica y se registra, enviando tanto sus datos personales, como la razón por la cual desea tener acceso a la organización. La lista de actividades que debe llevar a cabo un usuario será determinada por cada una de las áreas involucradas, las cuales especificarán el inicio de su flujo de trabajo a gestionar así como el fin de este tal como se ilustra en la Figura 3.14.

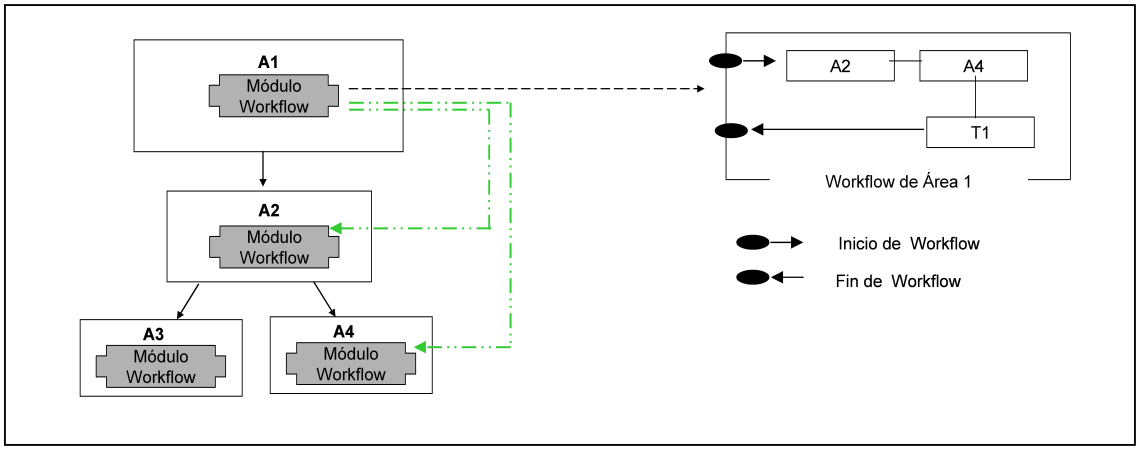

Figura 3.14: Flujo de trabajo en áreas jerárquicas

La clase `Workflow` que se encarga de crear la lista de actividades del usuario, así como de informar a las áreas involucradas del objetivo que el usuario persigue, para que cada área involucrada este en espera de la entrada del usuario y le estipule un flujo de trabajo específico.

En la Figura 3.15 se ilustra las clases correspondientes que se encargan de determinar las actividades del flujo de trabajo del usuario, las relaciones de las clases son las siguientes:

- Las clases `Workflow` y `ConjuntoActividades` mantienen una relación de composición con una cardinalidad `1...1`, debido que un flujo de trabajo solo puede tener un conjunto de actividades. Por otra parte un conjunto de actividades únicamente corresponden a un flujo de trabajo. El conjunto de actividades es determinado por el objetivo del usuario.

- Las clases `ConjuntoActividades` y `Actividad` tienen una relación "m a n". Un conjunto de actividades puede contener una o varias actividades y, a su vez, una actividad puede existir en varios conjuntos de actividades.

La clase `Workflow` está constituida por métodos que permiten construir el flujo de trabajo que el usuario nómada llevará a cabo dentro de una determinada área. Una simple actividad es modelada en la clase `Actividad`, mientras que la clase `ConjuntoActividades` guarda el grupo de actividades correspondientes al flujo de trabajo.

A continuación, se describe los atributos y métodos más relevantes de la clase `Workflow` la cual crea una instancia de las clases `ConjuntoActividades` y `Actividad`.

1. Atributos de la clase `Workflow`:

- `nombre`: define el nombre de un flujo de trabajo;
- `planactividades`: es un arreglo que almacena un conjunto de actividades; y
- `rolesvalidos`: determina los roles válidos para ejecutar un flujo de trabajo.

2. Métodos de la clase `Workflow`:

   - `agregaractividad()`: añade actividades al arreglo `planactividades`, instanciando la clase `ConjuntoActividades`, la cual a su vez crea una instancia de la clase `Actividad`;
   - `eliminaractividad()`: elimina una actividad del arreglo `planactividades`;
   - `asignaractividadparticular()`: asigna el flujo de trabajo a un rol, i.e., a un usuario le determina las actividades a realizar;
   - `desactivaractividad()`: desactiva el flujo de trabajo asignado por el método anterior;
   - `ejecutar()`: ejecuta el flujo de trabajo instanciando el método `ejecutarconjuntoactividades()` de la clase `ConjuntoActividades`;
   - `parar()`: detiene el flujo de trabajo en caso de ser necesario; y
   - `actividadactual()`: retorna la actividad actual que un usuario está llevando a cabo. El valor de esta actividad es enviado a las demás áreas involucradas para informarles que el usuario concluyó sus actividades en el área actual y que es momento de que acceda a la siguiente área.

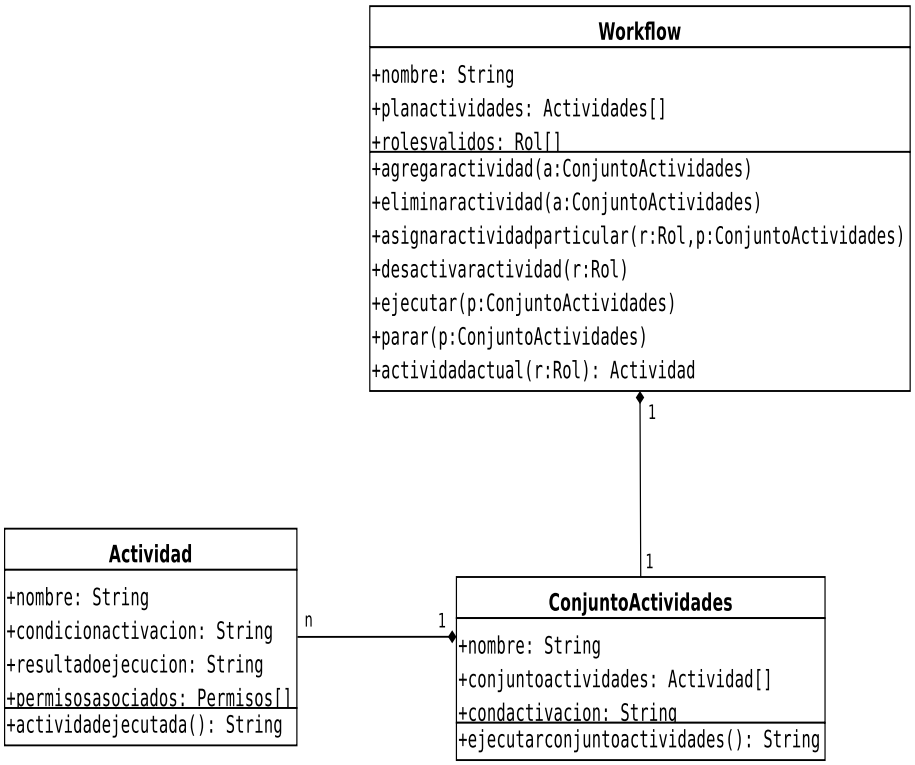

Figura 3.15: Clase `Workflow`, `ConjuntoActividades` y `Actividades`

Es importante mencionar que la clase `ConjuntoActividades` instancia a la clase `Actividad` para generar el conjunto de actividades que contendrá el flujo de trabajo correspondiente al objetivo del usuario. La clase `Actividad` asocia los permisos correspondientes a la actividad instanciando a la clase `Permiso`, mientras que el método `actividadejecutada()` se encarga de ejecutar la actividad. El valor que obtiene el atributo `resultadoejecucion`, determinará si se activa o no la siguiente actividad.

3.4.5 Clase `Comunicación`

La clase `Comunicación` se encarga de establecer una conexión entre las áreas involucradas en el cumplimiento total del objetivo del usuario. Está clase instancia la clase `Área` para obtener las áreas a las que puede acceder el usuario (ver Figura 3.16).

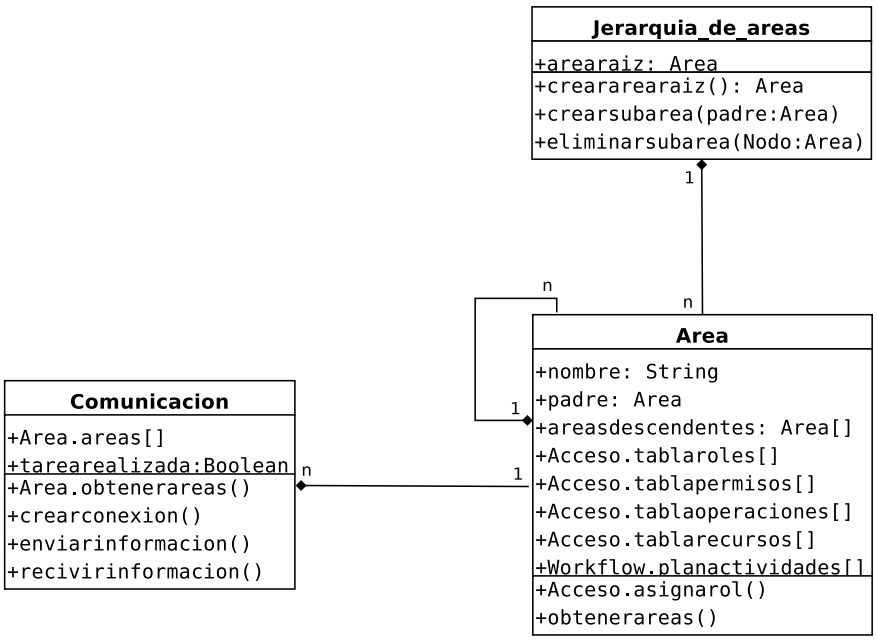

Figura 3.16: Clases que establecen la comunicación con las áreas involucradas

Entre las clases `Comunicación` y `Área` se define una relación de composición con cardinalidad de `1...n`, ya que una área tienen comunicación con las áreas involucradas. Por otro lado, las áreas involucradas solo se comunican con el área que las instanció.

1. Atributos de la clase `Comunicación`:

   - `Área.areas[]`: es un arreglo que almacena las áreas involucradas;
   - `tarearealizada`: determina si el usuario cumplió o no con sus actividades en las áreas involucradas.

2. Métodos de la clase `Comunicación`:

   - `Área.obtenerareas()`: crea una instancia de la clase `Área` para obtener las áreas involucradas y las guarda en el arreglo `areas[]`;

- `crearconexion()`: se encarga de establecer una conexión con las área involucradas;
- `enviarinformacion()`: se encarga de enviar la información de un usuario a las áreas involucradas; y
- `recibirinformacion()`: obtiene la información que las áreas involucradas envían.

3.5 Arquitecturas de distribución y de comunicación del sistema W-RBACSoft

En los últimos tiempos, la tecnología par a par (P2P) ha tenido un crecimiento y un impacto notable, lo cual hace que la gente especializada en el tema manifieste que esta tecnología se está convirtiendo en la sucesora del modelo clásico cliente/servidor. Este último tiene una arquitectura monolótica carente de soporte para la distribución de tareas, ya que generalmente todo el procesamiento es realizado en el servidor, mientras que el usuario no tiene ni siquiera la capacidad de efectuar operaciones simples, e.g., verificar que los campos de un formulario hayan sido proporcionados correctamente [Schoder and Fischbach, 2008].

Las redes P2P son utilizadas para muchos propósitos, tales como el uso compartido de archivos de texto, multimedia y datos en tiempo real (e.g., recursos, servicios e información). De acuerdo a su grado de centralización, la tecnología P2P se clasifica en: 1) arquitecturas híbridas con componentes centralizados, 2) puras o totalmente descentralizadas y 3) mixtas o semicentralizadas [Androutsellis-Theotokis and Spinellis, 2004].

P2P centralizadas

Constituyen la primera generación de redes P2P (ver Figura 3.17). Se basa en una arquitectura monolítica en la que todas las transacciones se realizan a través de un único servidor, que sirve de punto de enlace entre dos nodos. El servidor central guarda los registros de conexión de cada nodo (e.g., IP, puerto y ancho de banda) y de los tipos de recursos compartidos que proporciona. De esta manera, cuando un nodo requiere el recurso de otro nodo realiza la petición al servidor central, el cual la procesa y regresa la respuesta con la información necesaria para que se ponga en contacto con el nodo proveedor.

Este tipo de arquitectura reúne las siguientes características:

- cuando un nuevo nodo desea unirse a la red, este debe ponerse en contacto con el servidor central para reportarle el tipo de recursos que puede compartir;
- se rige bajo un único servidor que sirve como punto de enlace entre los nodos y como servidor de acceso al contenido, el cual distribuye el contenido a petición de los nodos; y
- todas las comunicaciones (peticiones de recursos y encaminamientos entre nodos) dependen exclusivamente del servidor.

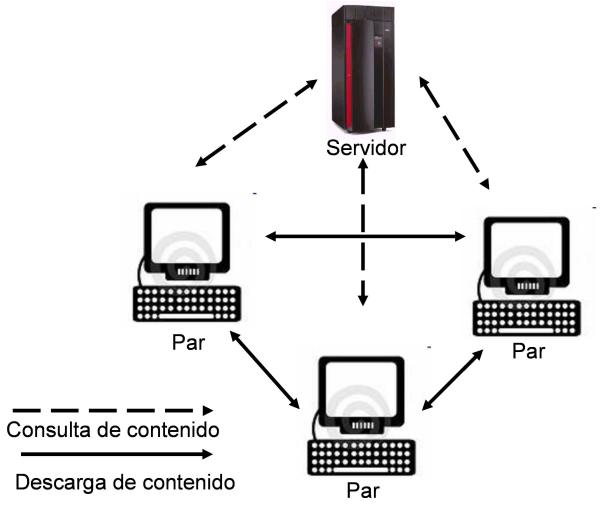

Figura 3.17: Arquitectura P2P centralizada

P2P puras o totalmente descentralizadas

Forman parte de la segunda generación de redes P2P, la cual es considerada como la más común (ver Figura 3.18). Las redes P2P puras no requieren de ningún tipo de administración, lo cual permite eliminar la necesidad de utilizar un servidor central. En otras palabras, esta tecnología utiliza los nodos como almacenes de información y como nodos de conexión, i.e., la comunicación se establece de par a par con la ayuda de un nodo intermediario.

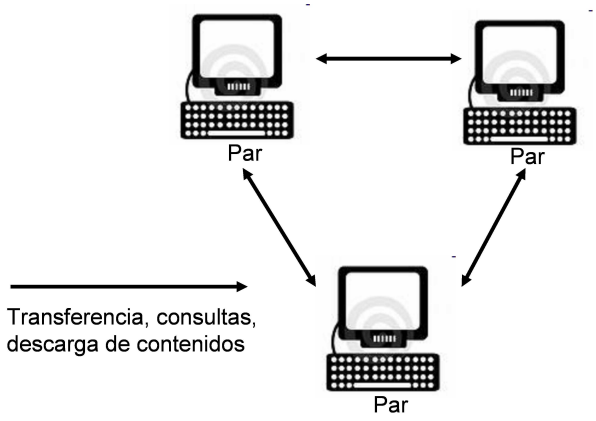

Figura 3.18: Arquitectura P2P pura o descentralizada

Este tipo de arquitectura reúne las siguientes características:

- los nodos actúan simultáneamente como cliente y servidor;
- no existe un servidor central que maneje las conexiones de red; y
- no hay un enrutador central que administre las direcciones de los nodos.

P2P híbridas, semi-centralizadas o mixtas

Constituyen la tercera generación (ver Figura 3.19). Este tipo de arquitectura es similar a la de componentes centralizados, ya que la interacción se lleva a cabo mediante un servidor central que sirve como *hub*. Este administra los recursos de ancho de banda, los enrutamientos y la comunicación entre los nodos, pero sin saber la identidad de cada nodo y sin almacenar información alguna, por lo que el servidor no comparte archivos de ningún tipo con ningún nodo. La tecnología P2P mixta permite la incorporación de más de un servidor que gestione los recursos compartidos. En caso de que él o los servidores se caigan, el grupo de nodos sigue en contacto a través de una conexión directa entre ellos, por lo que es posible seguir compartiendo y descargando información en ausencia de los servidores.

A estos servidores se les conoce como **superpar**. Ciertos nodos de la red son seleccionados como superpares para ayudar a administrar el tráfico dirigido hacia otros pares. Cada superpar mantiene un pequeño número de conexiones abiertas con los pares cliente.

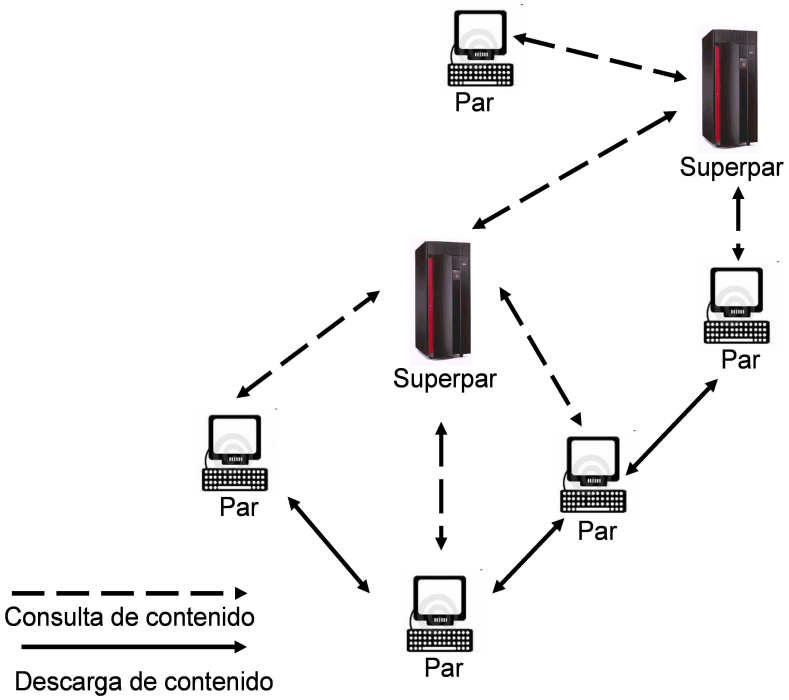

Figura 3.19: Arquitectura P2P mixta

Este tipo de arquitectura reúne las siguientes características:

- cuenta con un servidor central que contiene información en espera y responde a peticiones relacionadas con esta información; y
- los nodos son responsables de hospedar la información, ya que el servidor central no almacena ningún tipo de información.

3.5.1 Comunicación entre áreas autónomas

El sistema W-RBACSotf está compuesto de varios sub-componentes, los cuales están distribuidos en las áreas de una organización.

La comunicación entre áreas autónomas se basa en la tecnología P2P cuyo el elemento fundamental es el par, el cual se define como la unidad de procesamiento básico, capaz de desarrollar algún trabajo útil y de comunicar los resultados a otro nodo de la red [Tejedor and Jesús, 2006]. Se utilizó la arquitectura P2P pura para la comunicación entre las áreas que conforman una organización.

Esquema de comunicación P2P entre áreas autónomas

De acuerdo a la estructuración de las áreas que se describió en la sección 3.1, la comunicación entre ellas está restringida a una comunicación bidireccional entre nodos adyacentes (ver Figura 3.20). Cada área se comporta simultáneamente como cliente y servidor para poder enviar y recibir información cuando así se requiera.

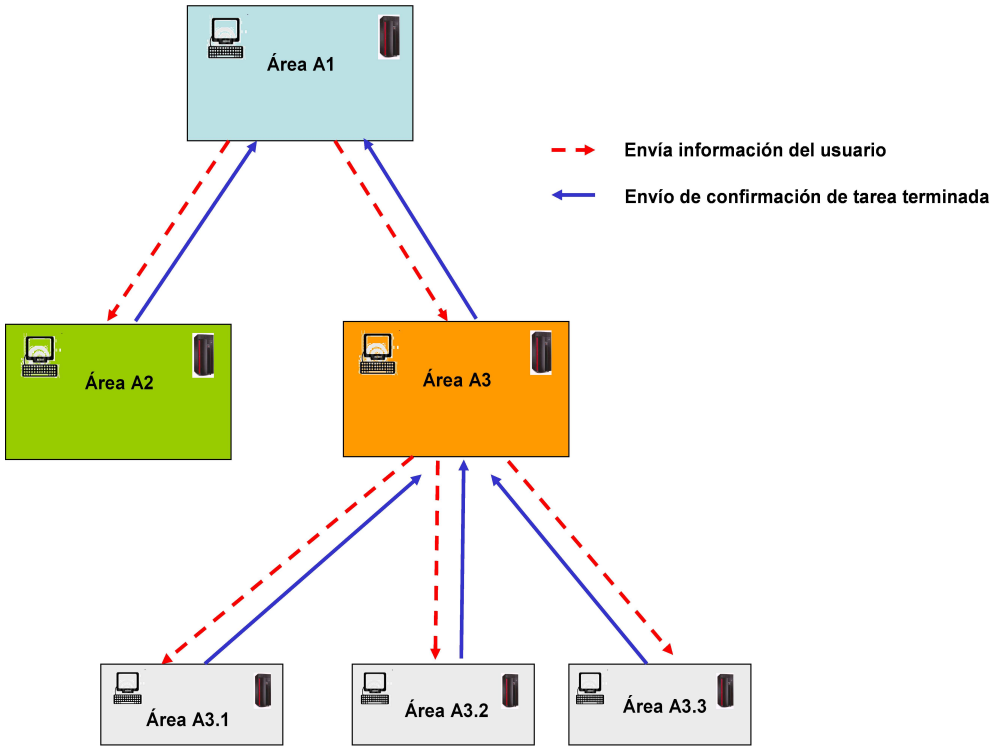

Figura 3.20: Esquema de comunicación entre áreas autónomas

Por ejemplo, suponga que un usuario nómada llega al área A1, donde se le asigna un rol y con base en su objetivo se determina el siguiente flujo de trabajo a realizar: 1) acceder al área A2 y 2) acceder al área A3. El área A1 crea una conexión con las áreas descendentes involucradas y les envía información del usuario que accederá a ellas. Cuando el usuario entra al área A2, recibe automáticamente sus nuevos permisos y actividades. El usuario realiza las actividades y al finalizarlas, el área A2 se comunica con el área A1 para notificarle que el usuario realizó satisfactoriamente sus actividades. De esta manera, el área A1 determina que ya llevo a cabo las actividades de uno de los flujos de trabajo

específicos. Finalmente, el usuario accede al área A3 donde recibe su nueva lista de actividades. Si el usuario requiere tener acceso a una de las áreas descendentes del área A3, e.g., el área A3.1, entonces el área A3 debería comunicarse con A3.1 para informarle que el usuario accederá a ella (ver Figura 3.21).

Una vez que el usuario realizó sus actividades dentro del área A3, esta debe comunicarse con su área ascendente y todas las áreas involucradas para informar que el usuario ha finalizado el flujo de trabajo general.

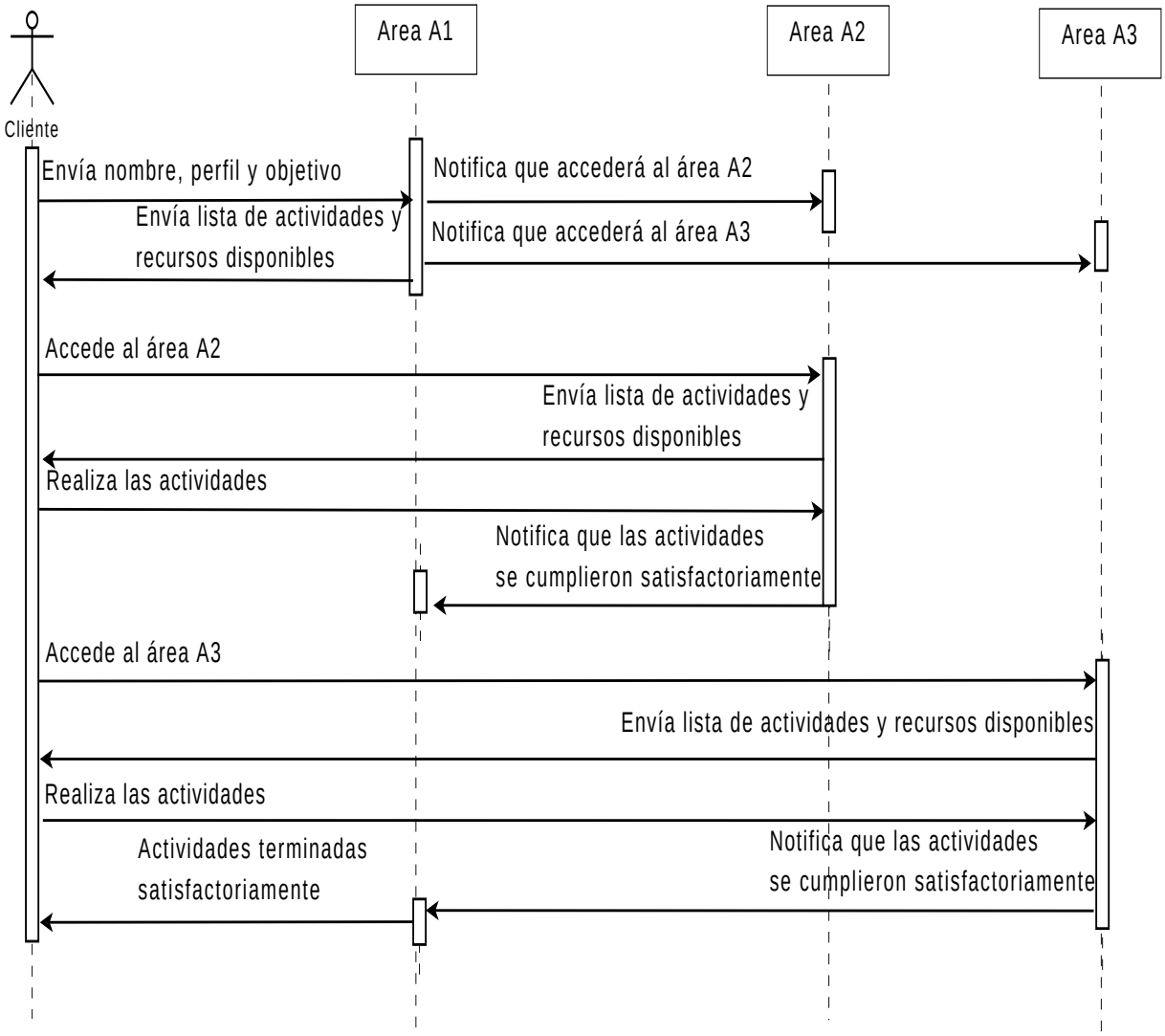

Figura 3.21: Diagrama de secuencia de la comunicación entre áreas autónomas

3.5.2 Comunicación cliente/servidor W-RBACSoft

El modelo cliente/servidor es una arquitectura modular basada en mensajes que está constituida por un cliente y un servidor. El cliente y el servidor actúan respectivamente como solicitante y proveedor de servicios [Edelstein, 1994].

La interacción entre estos actores se lleva a cabo de la siguiente manera: el cliente envía una solicitud al servidor, el cual la procesa y transmite al cliente una respuesta en forma de mensaje.

Las características de un cliente son:

- inicia solicitudes o peticiones, por lo tanto tienen el papel activo en la comunicación;
- espera y recibe las respuestas del servidor; y
- puede conectarse a varios servidores a la vez.

Las características de un servidor son:

- al ejecutarse se queda en espera de solicitudes de los clientes;
- tras la recepción de una solicitud, la procesa y luego envía la respuesta al cliente; y
- acepta conexiones de un gran número de clientes (en ciertos casos, el número máximo de peticiones puede estar limitado).

En función de los requerimientos necesarios para la comunicación entre el usuario y las áreas autónomas, se eligió el modelo cliente/ servidor, ya que el usuario solo se encarga de solicitar sus recursos y su flujo de trabajo correspondiente al área en la que se encuentra, mientras que las áreas se encargan de proporcionar la información solicitada por el usuario (ver Figura 3.22).

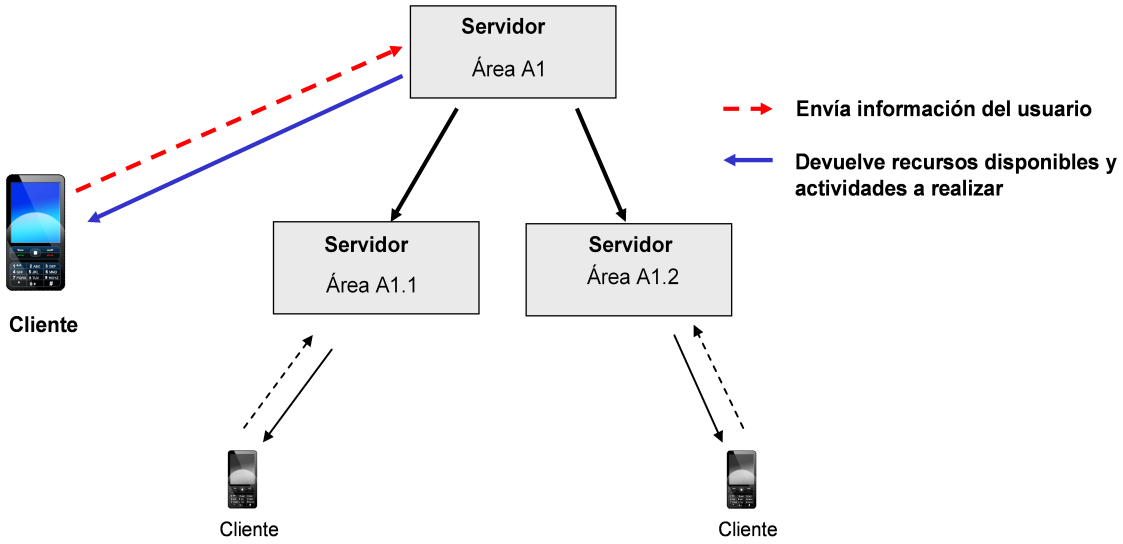

Figura 3.22: Comunicación entre el cliente y el servidor de un área autónoma

Cada área está en espera de la llegada del usuario nómada, el cuál a través de una aplicación cliente interactúa con el servidor del área donde se encuentra.

La aplicación cliente inicia la comunicación con el servidor del área. Esta aplicación cliente envía al servidor la información del usuario (nombre, perfil y objetivo). El servidor recibe la información y la almacena en su registro de usuarios. Posteriormente, el valor de la variable contextual `perfil` determina los permisos del usuario sobre los recursos del área. El valor de la variable contextual `objetivo` define el flujo de trabajo que el usuario realizará. Estos datos son enviados a la aplicación cliente (ver Figura 3.23).

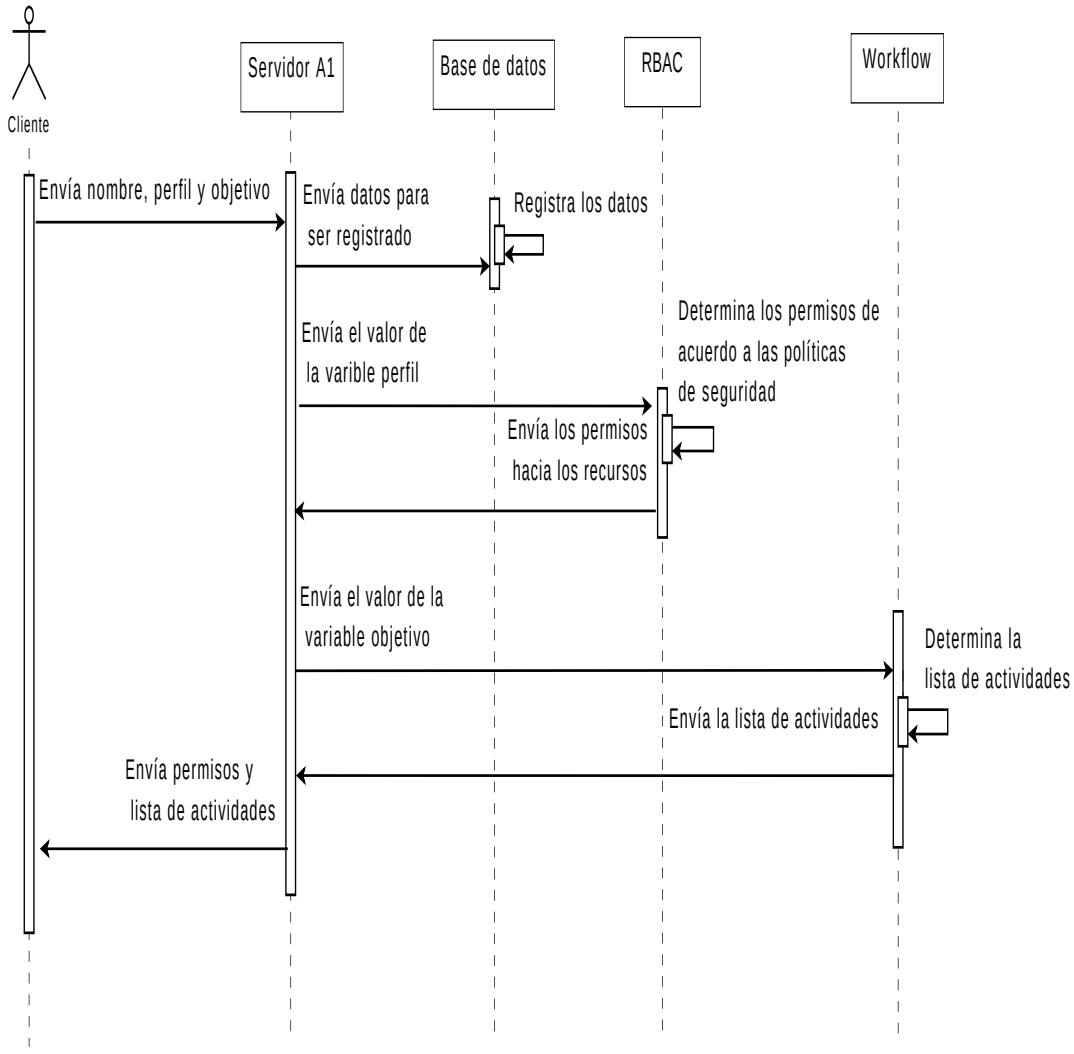

Figura 3.23: Diagrama de secuencia de la comunicación cliente/servidor W-RBACSoft

Cada vez que un usuario nómada accede a un área, se crea una comunicación cliente-servidor entre la aplicación cliente instalada en el dispositivo móvil del usuario y el servidor del área involucrada.

Capítulo 4

Implementación del sistema W-RBACSoft

El presente capítulo está estructurado en cinco partes. En primer lugar, se describe el lenguaje de programación que se uso en la implementación del sistema W-RBACSoft (cf. sección 4.1). A continuación, se describe el mecanismo para delimitar los rangos de coordenadas de las áreas de la organización (cf. sección 4.2). Posteriormente, se presenta la estructura de la base de datos que almacena la información de las áreas de la organización (cf. sección 4.3). Enseguida, se explica la implementación de los componentes más relevantes del sistema W-RBACSoft tales como: 1) la aplicación cliente, 2) el control de acceso, 3) los flujos de trabajo y 4) la comunicación entre el cliente y el servidor, así como la comunicación entre áreas (cf. sección 4.4). Finalmente, se presentan las pruebas realizadas al sistema W-RBACSoft (cf. sección 4.5).

4.1 Selección del lenguaje de programación

Una de las tareas fundamentales en el desarrollo del sistema W-RBACSoft es definir el lenguaje de programación más conveniente para su implementación. El sistema engloba dos ambientes con capacidades de programación distintas, i.e., una PC difiere del reducido entorno de programación de un dispositivo móvil.

El sistema propuesto en la presente tesis de maestría sigue el paradigma de programación orientada a objetos [Cook, 1990], el cual permite que los programas obtenidos sean más fáciles de escribir, mantener, y reutilizar. El lenguaje Java está diseñado como un lenguaje orientado a objetos que se ha convertido en el lenguaje de elección para implementar aplicaciones para entornos cada vez más complejos y basados en red. Java proporciona una colección de clases, que permite abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas. Las características que ofrece Java se ajustan a los requerimientos de implementación del sistema que sustenta la presente tesis de maestría. Para la implementación del cliente W-RBACSoft se disponía de un dispositivo móvil N95 con sistema operativo *Symbian S60* y para la implementación del servidor W-RBACSoft se utilizó una laptop *Acer Aspire 5050* y una *Dell Inspirion 1501*, ambas con sistema operativo Linux.

Herramienta de desarrollo de la aplicación cliente W-RBACSoft

La herramienta utilizada para la implementación de está aplicación es J2ME (*Java 2 Micro Edition*). La plataforma J2ME constituye una familia de especificaciones que definen varias versiones minimizadas de la plataforma Java. Estas versiones pueden ser utilizadas para programar aplicaciones ejecutables en dispositivos de mano; desde teléfonos móviles y PDAs hasta tarjetas inteligentes. Estos dispositivos tienen memoria limitada, cuentan con pantallas muy reducidas, su nivel de procesamiento es limitado y no necesitan todo el soporte que brinda J2SE (plataforma estándar de Java empleada en sistemas de escritorio y servidores) [de Malaga, 2003]. Por lo tanto, las interfaces de usuario para el dispositivo móvil N95 se realizaron con J2ME utilizando el IDE NetBeans 6.1.

Herramienta de desarrollo para el servidor W-RBACSoft

Como se mencionó en la sección 3.1, la propuesta de este trabajo de investigación es la administración basada en áreas de una organización. Cada área cuenta con un servidor W-RBACSoft, el cual se encarga de administrar su información, recursos y actividades de manera autónoma. En la implementación del sistema W-RBACSoft se contemplaron dos áreas: 1) `Recepción` y 2) `Departamento de Computación`, las cuales deben ser capaces de interactuar con el usuario y entre ellas mismas.

La herramienta de programación utilizada para el desarrollo del servidor W-RBACSoft es el lenguaje java. Una de las ventajas de emplear este lenguaje es que puede ser ejecutado tanto en el sistema operativo Linux como en Windows, siempre y cuando se tenga instalada la máquina virtual de Java [Gosling et al., 2005].

4.2 Mapa de ubicación de las áreas

La implementación del sistema W-RBACSoft requiere definir el rango de coordenadas de cada área que compone la organización. Por esta razón, se determinaron primeramente las áreas que integran la institución, donde se implemento el sistema. La institución está compuesta de las siguientes áreas: 1) `Recepción`, 2) `Departamento de Computación` y 3) `Departamento de Mecatrónica`, de las cuales solo se contemplaron dos de ellas para la implementación del sistema (`Recepción` y `Departamento de Computación`).

Rango de coordenadas de las áreas

Para determinar el rango de coordenadas de las áreas se diseño el mapa de la institución, y se identificó las áreas y sus secciones. Con base en las dimensiones del edificio de la institución y sus divisiones físicas, se determinó el rango de coordenadas de cada área, las cuales fueron almacenadas en la base de datos del área para conocer la ubicación de cada una de ellas dentro de la institución, con estos rangos de coordenadas ya puede identificarse la ubicación del usuario (ver Figura 4.1).

Para cálcular de las dimensiones de cada área se tomó como referencia los ejes de coordenadas X, Y. Con base en las coordenadas (0, 0) los valores de X corresponden a los valores que van hacia la derecha del mapa del edificio, mientras que los valores Y pertenecen a las medidas que van hacia la parte superior del edificio.

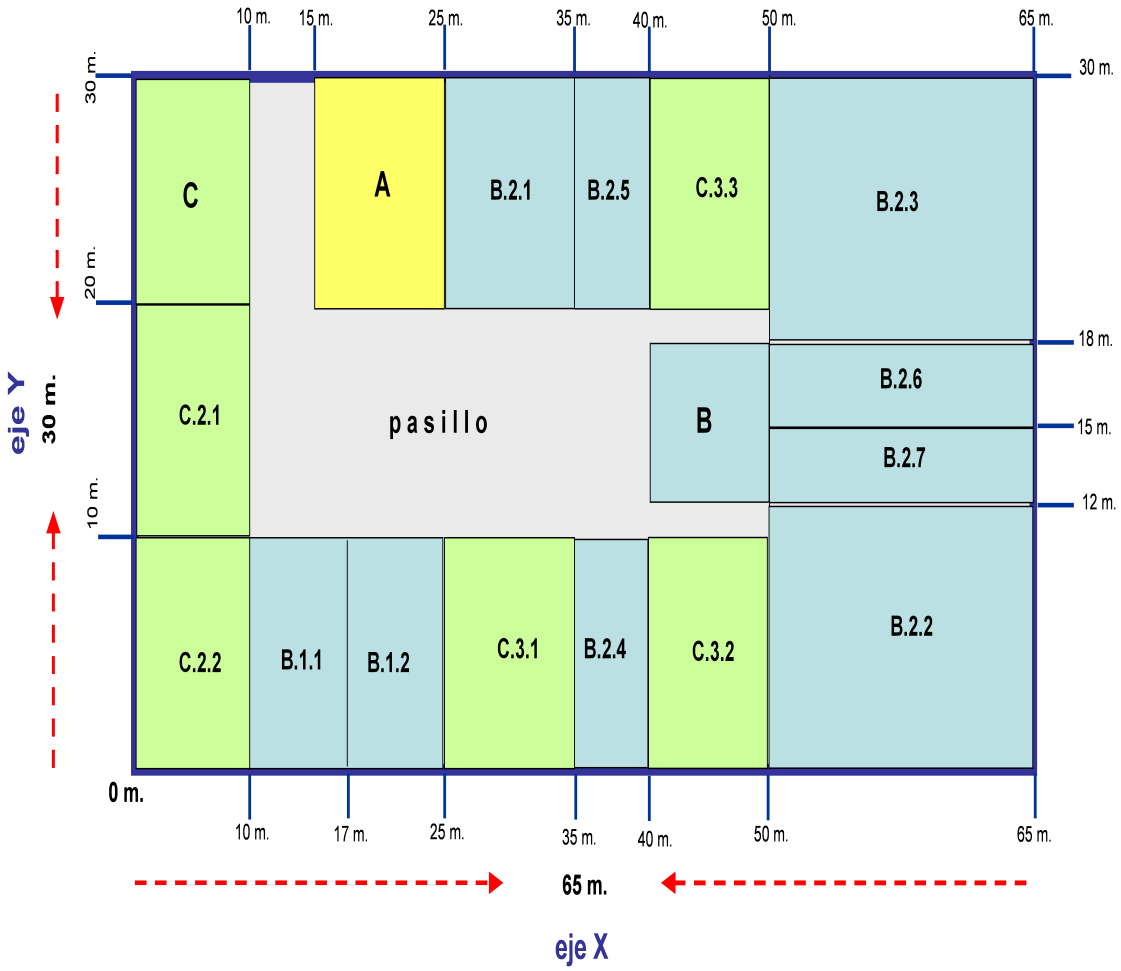

Figura 4.1: Mapa de coordenadas de la institución

El cálculo para determinar el rango de coordenadas de cada área es el siguiente:

área A(`rangoInicial`, `rangoFinal`)

donde:

- área A, corresponde a un área de la institución;
- `rangoInicial`, representa a las coordenada en X y Y, donde inicia el límite del área A; y
- `rangoFinal`, representa a las coordenada en X y Y, donde finaliza el límite del área A.

Por ejemplo:

área `Recepción`((15, 20),(25, 30))

donde:

El límite de inicial del área `Recepción` está determinado por X=15 y Y=20, mientras que el límite final está dado por X=25 y Y=30.

Suponga, que se quiere conocer la ubicación del usuario dentro de la institución. Las coordenadas de posición del usuario son X=20 y Y=25, de acuerdo al mapa de coordenadas de la institución (ver Figura 4.1) se puede determinar que se encuentra en el área A (área `Recepción`).

Para identificar en que lugar de la institución se encuentra un usuario, se debe realizar una comparación de las coordenadas del usuario con los rangos de los valores comprendidos por cada una de las áreas.

El procedimiento 1 "Localización" determina en que área se encuentra un usuario. Los rangos de cada área ya se están definidos. Para llevar a cabo dicha comparación se toma el valor de la posición en X del usuario y se compara con el rango de valores en X correspondientes al área. De igual manera se compara la posición Y del usuario con el rango de valores en Y del área.

Algoritmo 1 Localización

Entrada: X, Y
Salida: $IdArea$

```
1: si (15 ≥ X ≤ 25) y (20 ≥ Y ≤ 30) entonces
2:    área = id_R                                   // Recepción
3: si no, si (40 ≥ X ≤ 50) y (12 ≥ Y ≤ 18) entonces
4:    área = id_DC                                  // Departamento de Computación
5: si no, si (0 ≥ X ≤ 10) y (20 ≥ Y ≤ 30) entonces
6:    área = id_DM                                  // Departamento de Mecatrónica
7: si no
8:    área = id_Pas                                 //Pasillo
9: fin si
```

Descripción del procedimiento 1 correspondiente a la localización:

1. al recibir las coordenadas X y Y del usuario, compara si los valores de X y Y están dentro del rango de las coordenadas del área de `Recepción`;
2. si la condición se cumple, devuelve el id del área;
3. en caso contrario, compara las coordenadas X,Y del usuario con el rango de coordenadas del `Departamento de Computación`;
4. si la condición se cumple, devuelve el id del área;
5. en caso contrario compara las coordenadas X,Y del usuario con el rango de coordenadas del área `Departamento de Mecatrónica`;
6. si la condición se cumple, devuelve el id del área; y

7. si ninguna de las condiciones anteriores se cumple; devuelve un valor por omisión, el cual puede representar el pasillo de la organización.

La tabla 4.1 muestra los datos que están almacenados en la base de datos de cada área tales como: identificador del área, nombre del área, rango de coordenadas y dirección IP (e.g., id1, Departamento de Computación, 12,34 - 14,20 y 192.168.45.56). Después de llevar a cabo el procedimiento anterior, se realiza una consulta a la base de datos para obtener el nombre del área en la que se encuentra el usuario.

Idarea	**Área**	**Rango de Coordenadas**	**IP**
Id1_R	Recepción	15,20 - 25,30	192.168.50.1
Id2_DC	Departamento de Computación	40,12 - 50,18	192.168.50.20
Id3_DM	Departamento de Mecatrónica	0,20 - 10,30	192.168.50.31
.	.	.	.
.	.	.	.
.	.	.	.
n áreas	.	.	.

Tabla 4.1: Datos correspondientes de cada área

El rango de coordenadas de cada área también permite determinar con qué servidor se conectará el usuario, ya que se utilizó una dirección IP (*Internet Protocol*) estática para el servidor de cada área (ver Figura 4.2).

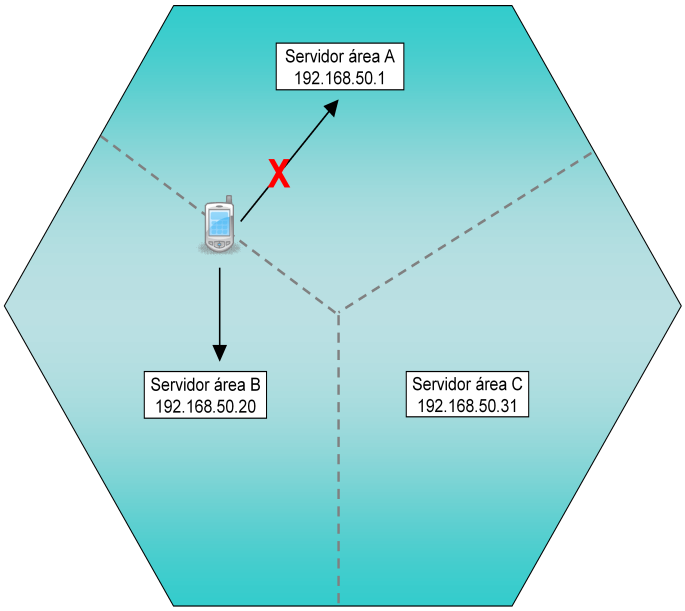

Figura 4.2: Dirección IP de cada área que compone la institución

4.3 Ubicación de la información de las área

Una colección de datos normalmente es denominada base de datos, la cuál contiene información relevante de una organización. El objetivo principal de las bases de datos es proporcionar una forma de almacenar y recuperar la información de manera que sea tanto práctica como eficiente. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para manipular la información [Silberschatz et al., 2002].

Cada área se encarga de almacenar toda la información que la integra, tal como: subáreas, recursos, permisos y usuarios. El servidor W-RBACSoft es responsable de interactuar con la información almacenada en la base de datos, realizando consultas e ingresando información a ella.

Estructura de la base de datos

La estructura de la base de datos de cada área adopta el modelo relacional, el cual se encarga de relacionar los datos unos con otros. El modelo relacional está basado en registros; se denominan así porque la base de datos está estructurada en registros de varios tipos. Cada tabla contiene registros de un tipo particular, mientras que estos definen un número fijo de campos, o atributos. Las columnas de la tabla corresponden a los atributos del tipo de registro [Silberschatz et al., 2002].

La Figura 4.3 ilustra la estructura que tiene la base de datos de cada una de las áreas de la organización. Cada tabla está compuesta por varias columnas, y cada columna tiene un nombre único respectivamente el cual facilita la relación entre tablas. Las tablas AreaN1, AreaN2 y AreaN3 almacenan la información correspondiente a cada una de ellas. Estas tablas estan relacionadas, con la finalidad de adoptar una estructura jerárquica.

- las tablas `AreaN1` y `AreaN2` mantienen una relación con cardinalidad de `1...n`, está relación define una estructura jerárquica donde la tabla `AreaN1` almacena su propia información y a su vez representa las áreas padre del las áreas almacenadas en la tabla `AreaN2`.

- las tablas `AreaN2` y `AreaN3` mantienen una cardinalidad de `1...n`, las áreas de la tabla `AreaN2` almacena su propia información y representan las áreas padre de las áreas almacenadas en la tabla `AreaN3`.

- La tabla `RecursosA_N1` mantienen una cardinalidad `n...1`, está tabla almacena todos los recursos con los que cuenta una determinada área.

- la tabla `RolesA_N1` almacenan los roles válidos dentro de un área.

- las tablas `Usuario_externo` y `Usuario_interno` se relacionan con la tabla `RolesA_N1` con la finalidad de determinar qué rol le corresponde a un usuario y a su vez define que permisos (operaciones a los recursos) le corresponden. Por esta razón, la tabla `RolesA_N1` se relaciona con la tabla `Permisos` teniendo una cardinalidad `1...n`.

- las tablas `Permisos` y `RecursosA_N1` se relacionan con una cardinalidad de n...m ya que varios permisos pueden ser asignados a varios recursos.

Cabe mencionar que, con base en el número de áreas que componen una organización, las relaciones entre áreas pueden crecer, e,g., si las áreas almacenadas en la tabla AreaN3 tuvieran áreas subyacentes, se generaría otra tabla para poder almacenarlas. La estructura de la base de datos es la misma para cada una de las áreas. Sin embargo, cada área guarda su propia información en su base de datos correspondiente.

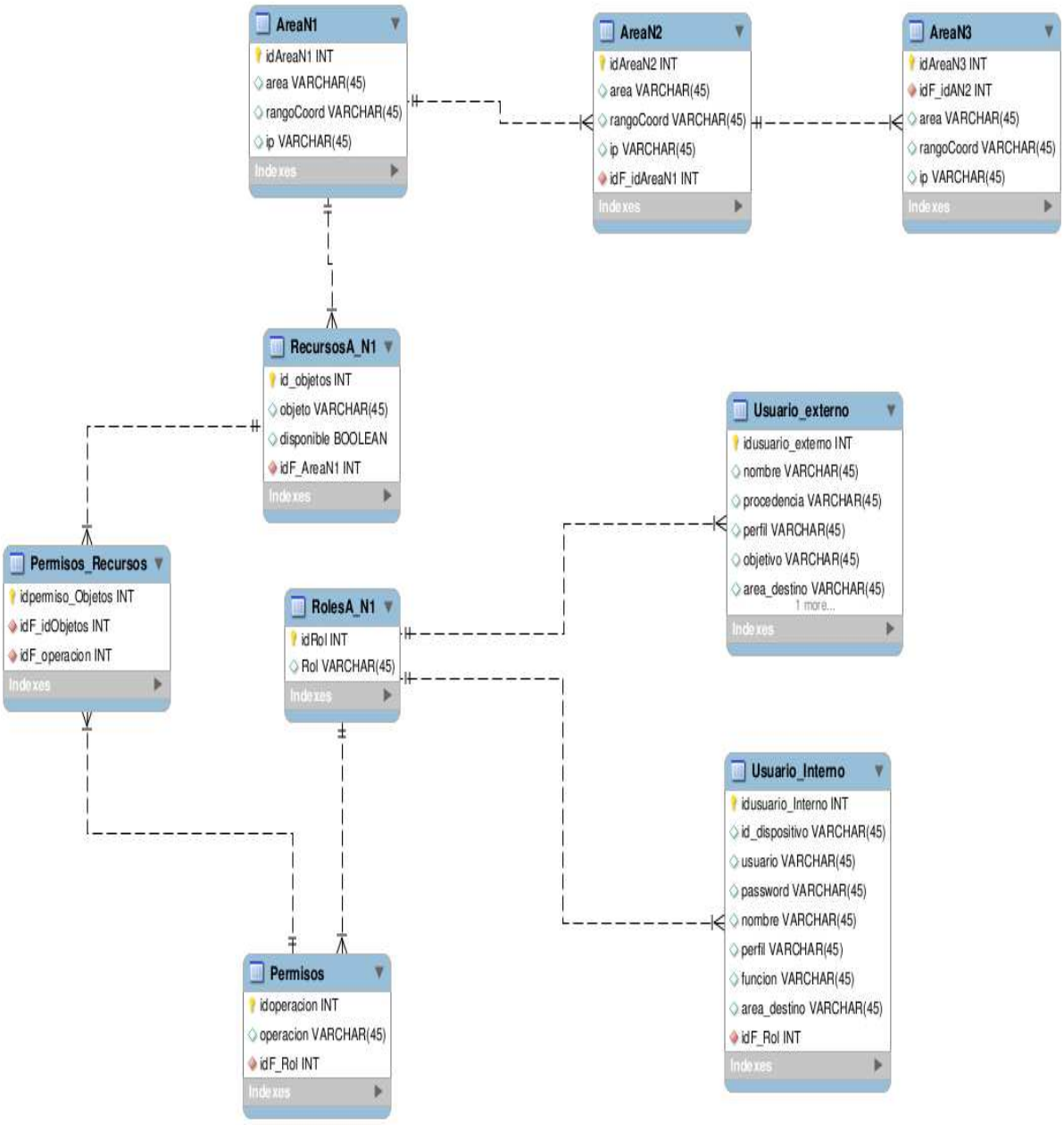

Figura 4.3: Estructura de la base de datos

El sistema manejador de base de datos utilizado para el almacenamiento de la información de las áreas es PostgresSQL 8.3. Es un software de distribución libre el cual es publicado bajo la licencia BSD (*Berkeley Software Distribution*). PostgresSQL es un potente

motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a las de muchos gestores de bases de datos comerciales. Es más completo que MySQL, ya que permite métodos almacenados, restricciones de integridad y vistas, aunque en las últimas versiones de MySQL se han hecho grandes avances en ese sentido [Worsley and Drake, 2002].

4.4 Implementación de componentes relevantes

En la sección 3.3.1 se describe la implementación de cada uno de los componentes del sistema W-RBACSoft. Esta sección se divide en tres subsecciones: 1) la aplicación cliente W-RBACSotf, 2) el servidor W-RABCSoft y 3) la comunicación entre los servidores W-RBACSotf.

4.4.1 Aplicación cliente W-RBACSoft

La aplicación cliente W-RABSoft se refiere a las interfaces que se presenta al usuario en el dispositivo móvil N95. Está aplicación interactúa con el servidor W-RABCSoft de cada área para enviar y recibir información correspondiente al usuario.

Interfaces de aplicación cliente W-RBACSoft

Como se mencionó en la sección 4.1.1 la implementación de esta parte de W-RBACSoft, fue desarrollada con el lenguaje orientado a dispositivos móviles denominado lenguaje J2ME. Cabe mencionar que las interfaces de usuario están orientadas a dos tipos de usuarios: 1) usuarios externos y 2) usuarios internos (ver Figura 4.4).

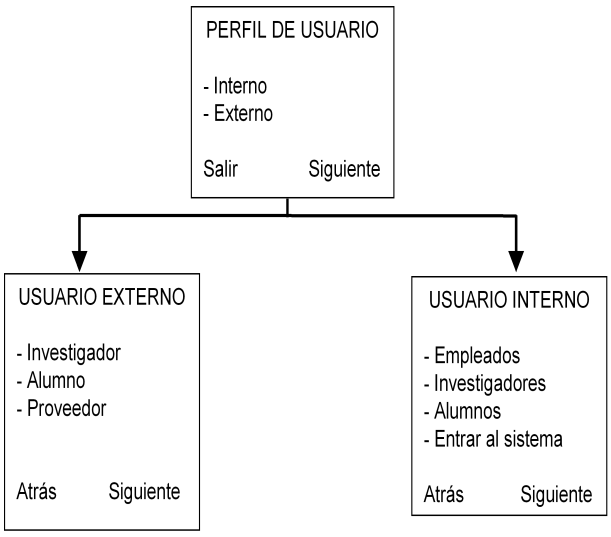

Figura 4.4: Interfaz para la selección de tipo usuario

Los usuarios externos son ajenos a la organización, lo que implica que desconozcan el proceso que deben llevar a cabo para realizar una determinada tarea. Por está razón, el sistema W-RBACSoft les proporciona un flujo de trabajo correspondiente y permisos sobre los recursos.

El usuario interactúa con la aplicación cliente W-RBACSoft para llevar a cabo su registro proporcionando sus datos (nombre, procedencia, perfil, objetivo y área destino). La interfaz que se muestra al usuario para realizar su registro es amigable, ya que basta con seleccionar la información adecuada. Los únicos datos proporcionados por el usuario son: su nombre, su procedencia y su área destino; es así como el usuario se registra en el servidor W-RBACSoft del área donde se encuentra (ver Figura 4.5).

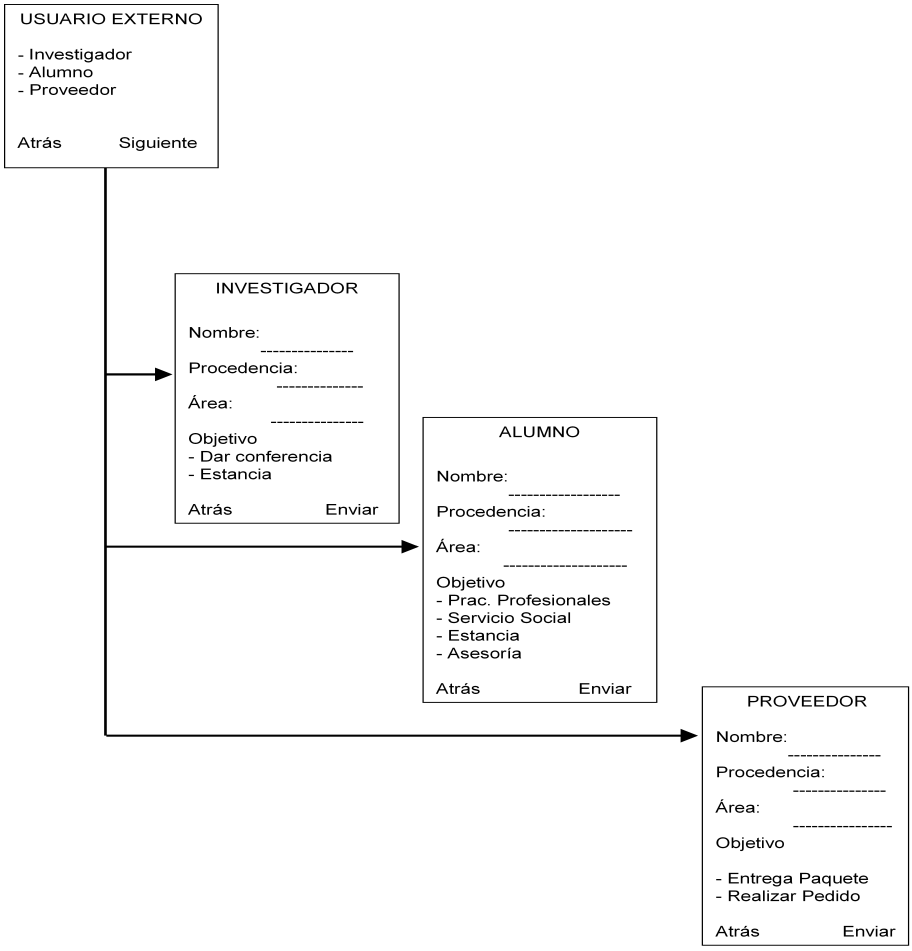

Figura 4.5: Interfaces de usuario para el registro de los datos de un usuario externo

Después de registrarse, el usuario obtiene en su dispositivo móvil N95 su información correspondiente tal como recursos y flujo de trabajo. La Figura 4.6 ilustra los componentes del cliente W-RABCSoft, así como la interacción con el servidor W-RBACSoft del área. Cuando el servidor W-RBACSoft envía la información a la aplicación cliente W-RBACSoft, esta la clasifica para mostrarla de manera ordenada al usuario. Las interfaces de usuario que son mostradas en la pantalla del dispositivo móvil se clasifican en "permisos"' y "flujo de trabajo". En la pantalla de "permisos" se muestran todos los recursos disponibles y las operaciones que puede realizar sobre ellos. En la pantalla de "flujo de trabajo" muestra la lista de actividades que el usuario debe realizar.

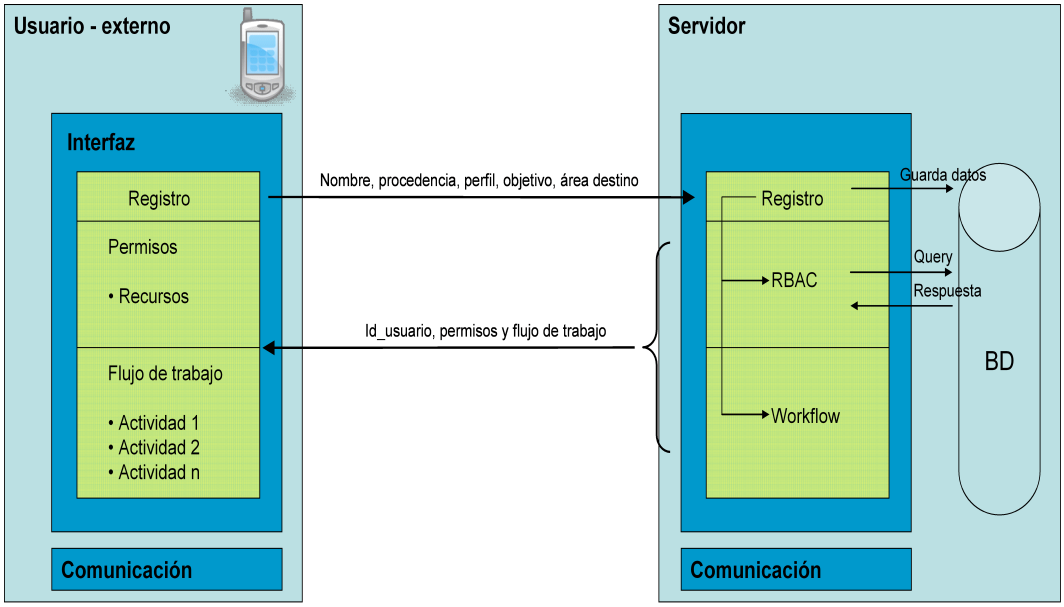

Figura 4.6: Componentes activos para el usuario externo

Los usuarios internos corresponden al personal que trabaja dentro de la organización. A este tipo de usuarios, el servidor W-RBACSoft solo se encarga de proporcionarles sus respectivos permisos sobre los recursos.

El sistema se limita a ofrecer flujos de trabajo a este tipo de usuarios ya que sus tareas son muy dinámicas y cambiantes, e.g., la secretaria de una institución llega a su entorno de trabajo. Primeramente debe llevar a cabo las inscripciones de los alumnos, pero en ese momento llega el coordinador y le pide que realice una solicitud al Departamento de Servicios Escolares. Por lo tanto, la secretaria debe interrumpir la primer tarea para realizar la tarea que le encomendo el coordinador. Si por algún motivo se le asignará una nueva tarea urgente, entonces debe interrumpir la tarea que este realizando en ese momento para resolver la de mayor prioridad y finalmente deberá ir retomando cada una de las tareas que dejo pendientes.

Con el ejemplo anterior podemos ver que un usuario interno no tiene definidas las tareas que realizará durante el día, estas tareas se pueden ir determinando en base a los objetivos que el entorno vaya requiriendo. Por esta razón queda fuera del alcance del sistema W-RBACSoft ofrecer flujos de trabajo a estos usuarios ya que su entorno de trabajo es muy cambiante.

Este tipo de usuario debe llevar a cabo su registro cuando interactúa por primera vez con el sistema W-RBACSoft. Al igual que los usuarios externos, los usuarios internos proporciona sus datos personales (nombre, perfil, función que desempeña y el área donde la desempeña) (ver Figura 4.7). Cuando se realiza el registro, el servidor W-RBACSoft devuelve su nombre de usuario, contraseña y sus respectivos permisos sobre los recursos.

El usuario no requiere navegar entre una pantalla y otra de la aplicación cliente W-RBACSoft ya que, como se muestra en la Figura 4.8, el usuario solo se encarga de realizar su registro en el sistema W-RBACSoft y posteriormente interactúa con la interfaz permisos (los permisos son enviados por el servidor).

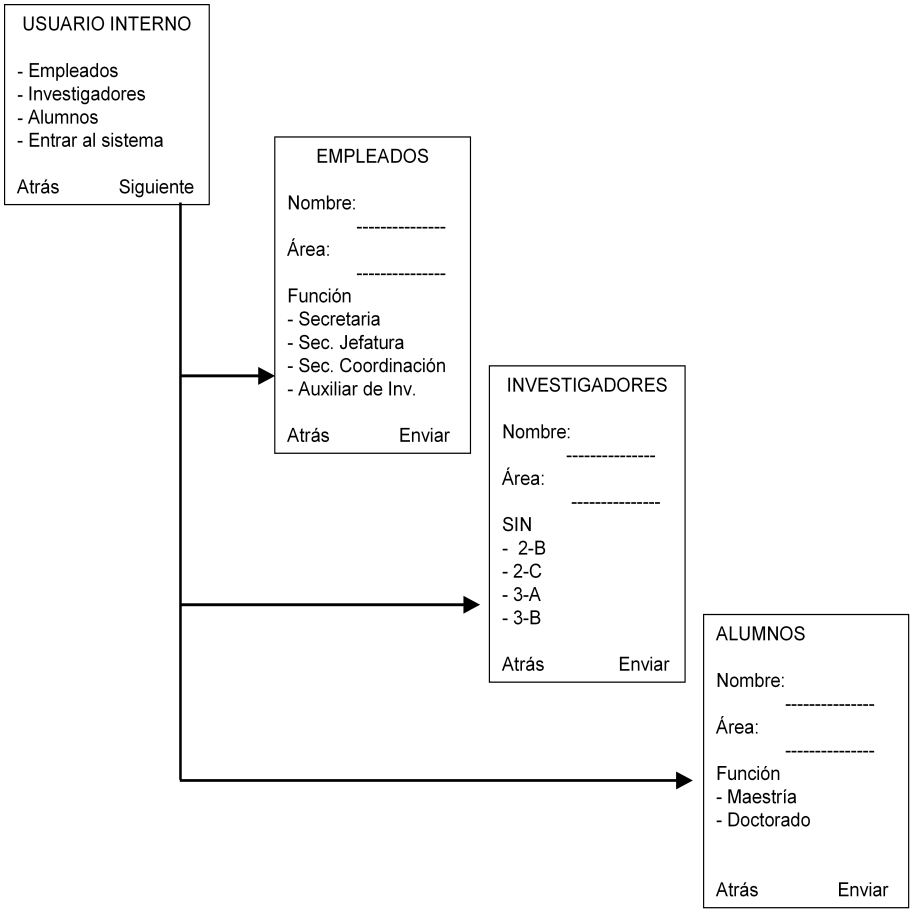

Figura 4.7: Interfaces de usuario para el registro de los datos de un usuario interno

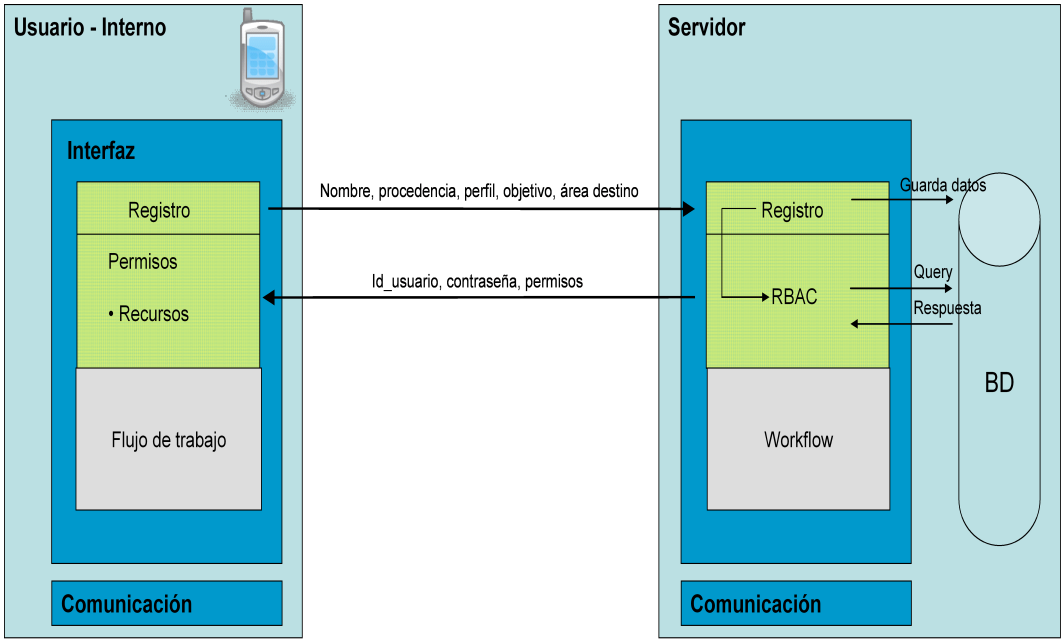

Figura 4.8: Componentes activos para un usuario interno

El procedimiento 2 "Selección de interfaz de usuario", describe de manera estructurada la forma en cómo un usuario navega en las interfaces de usuario de la aplicación cliente W-RBACSoft.

Este procedimiento está enfocado principalmente al registro del usuario al sistema W-RBASoft, primeramente se debe determinar que tipo de usuario va a realizar el registro, ya que existen dos tipos de usuarios, internos y externos. Después de seleccionar que tipo de usuario se registrará, el sistema le proporcionara su respectivos permisos.

Algoritmo 2 Selección de interfaz de usuario

Entrada: $nombre, procedencia, perfil, objetivo, funcion, areadestino$
Salida: $IdArea$, $recursos$, $flujodetrabajo$

1: **si** usuario externo **entonces**
2: realiza registro de datos // nombre, procedencia, perfil, objetivo, área destino
3: da clic en enviar //los datos son enviados al servidor W-RBACSoft
4: usuario espera notificación de registro
5: **si** registro exitoso **entonces**
6: aplicación cliente W-RBACSoft muestra en pantalla las opciones: 1) "permisos" y 2) "flujo de trabajo"
7: **si** usuario selecciona opción "permisos" **entonces**
8: aplicación cliente W-RBACSoft muestra en pantalla los recursos disponibles
9: visualiza los permisos del recurso seleccionado
10: **fin si**
11: **si** usuario selecciona opción "flujo de trabajo" **entonces**
12: aplicación cliente W-RBACSoft muestra en pantalla lista de actividades
13: usuario selecciona cada actividad terminada y envía notificación al servidor W-RBACSoft
14: **fin si**
15: **fin si**
16: **si** registro no exitoso **entonces**
17: regresa a paso 2
18: **fin si**
19: **fin si**
20: **si** usuario interno **entonces**
21: reliza registro de datos // nombre, perfil, función, área destino
22: da clic en enviar //los datos son enviados al servidor W-RBACSoft
23: usuario espera notificación de registro
24: **si** registro exitoso **entonces**
25: aplicación cliente W-RBACSoft muestra en pantalla los recursos disponibles al usuario
26: visualiza los permisos del recurso seleccionado
27: **fin si**
28: **fin si**
29: **si** registro no exitoso **entonces**
30: regresa a paso 2
31: **fin si**

4.4.2 Control de acceso

El Control de Acceso Basado en Roles, mejor conocido por sus siglas en inglés como RBAC (*Role Based Access Control*) es una tecnología que satisface las principales necesidades en cuanto a control de accesos se refiere. El control de acceso actualmente forma parte de un conjunto de políticas. Los sistemas con RBAC ofrecen una administración de seguridad mucho más confiable, debido a la asignación de roles para determinar los permisos.

Cuando algún usuario quiere acceder a un conjunto de datos, el servidor busca la condición que especifique si ese usuario tiene permitido o no accederlos. Si tiene permiso, el usuario podrá acceder a la información; de lo contrario el usuario no puede realizar ningún tipo de acceso. Por ejemplo, si un usuario con perfil `proveedor` desea entrar a la organización, el servidor buscará una regla que determine si este perfil puede tener acceso; si la encuentra, se le permitirá el acceso, de lo contrario se le negará dicho acceso [Ibarra and Mañas, 2003].

El sistema W-RBACSoft se encarga de implementar las políticas que administran los privilegios de los usuarios dentro de la organización. Este componente actúa en el momento en que el usuario accede a las áreas que componen la organización, el control de acceso utiliza como referencia el identificador de permisos asignados a los usuarios, los cuales determinan las operaciones que podrá realizar sobre los recursos. Los permisos se encuentran asociados a los roles, cuyos miembros son los usuarios.

Las entidades principales que se contemplaron en el control de acceso son:

1. *usuario*: es la persona que interactúa con el sistema W-RBACSoft;
2. *proceso*: representa un proceso que actúa en nombre del usuario, i.e., aplicación cliente que se ejecuta en el dispositivo del usuario;
3. *recurso*: es cada uno de los recursos de la organización;
4. *operación*: es la acción a realizar dentro de un proceso; y
5. *permiso*: es la acción aplicada a un recurso.

Estas entidades son las que se tomaron en cuenta en el sistema W-RBACSoft para determinar las reglas de las políticas de la organización y adaptarlas. Dichas entidades se precisaron con base en la descripción formal del modelo RBAC [Ferriolo and Kuhn, 1992]. Tales reglas son las siguientes:

Para cada usuario, el rol activo es el que tiene asignado a su proceso cliente actual:

$$Rol\_activo(p{:}proceso) = \{rol\ activo\ en\ el\ proceso\ p\}$$

Un proceso puede tener asignado uno o más roles:

$$Rol\_autorizado(p{:}proceso) = \{roles\ autorizados\ en\ el\ proceso\ p\}$$

Cada rol puede ser autorizado para llevar a cabo uno o más permisos:

$$Permiso\_atorizado(r{:}rol) = \{permisos\ autorizados\ para\ el\ rol\ r\}$$

Al proceso de cada usuario se le asignan permisos, por lo tanto la condición *Realiza_permisos(p, pe)* se define:

$$Realiza\_permisos(p{:}proceso,\ pe{:}permiso)$$

La condición anterior será *verdadera* si el proceso p tiene asignado el permiso *pe*.

Con base en las reglas anteriores se definen dos reglas básicas e importantes en el sistema W-RBACSotf:

1. **Asignación de rol**: un usuario inicia un proceso solo si tiene asignado un rol.

   $$\forall p{:}proceso,\ r{:}rol, (Asigna\_rol(p,\ r) \Rightarrow Rol\_activo(p) \neq 0)$$

   Los privilegios de un usuario se determinan por los permisos que le son asignados a través de un rol.

2. **Autorización de permisos**: un proceso puede tener un permiso solo si el permiso es autorizado en el rol activo del proceso.

   $\forall p{:}proceso, pe{:}permiso, (Realiza\_permisos(p,pe) \Rightarrow pe \in Permiso\_atorizado(Rol\_activo(proceso)))$

4.4.3 Flujos de trabajo (*Workflow*)

Una aplicación de flujos de trabajo (*workflow*) automatiza las secuencia de acciones, actividades o tareas utilizadas para la ejecución de un proceso, incluyendo el seguimiento del estado de cada una de sus etapas y la aportación de las herramientas necesarias para gestionarlo [Gerónimo and Canseco, 2002], los elementos clave que forman a un proceso son los que se ilustran en la la Figura 4.9.

Los aspectos importantes para la descripción de los flujos de trabajo están basados en:

1. Identificación de actividades:
   - módulos que deben lanzar las actividades;
   - acciones automáticas a realizar; e
   - información que se va a presentar al usuario.
2. Aplicaciones externas:
   - identificación de eventos; y
   - módulos que deben originar los eventos.

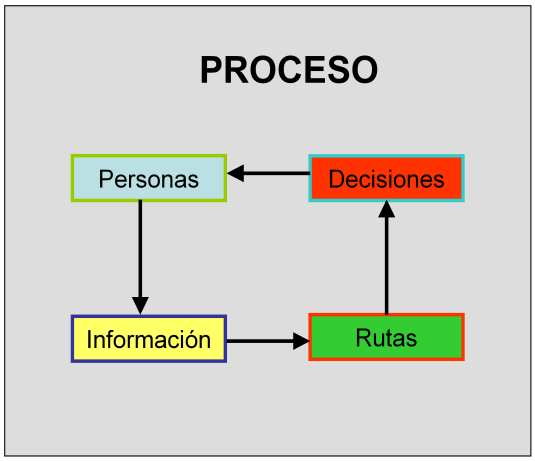

Figura 4.9: Elementos clave de un proceso

Framework EMF

La implantación de la tecnología de flujos de trabajo en las organizaciónes debe considerarse más como un proyecto de gestión que como un sistema informático, lo que conlleva a un esfuerzo de análisis previo de la situación de la organización. En consecuencia, se tiene que estudiar la circulación de los documentos, su naturaleza y tratamiento, así como la asignación de normas y tareas a los usuarios, y el orden de ejecución de estas tareas.

La implementación de este componente está compuesta de dos aspectos importantes que permiten la creación de los flujos de trabajo correspondientes al entorno de la organización. Nos referimos al modelo del diseño de los flujos de trabajo para cada objetivo del usuario y el modelo en tiempo de ejecución.

Para la construcción de los flujos de trabajo se requirío el apoyo de una herramienta, la cual fue proporcionada por el entorno de desarrollo del IDE Eclipse. Este entorno ofrece un *framework* EMF (*Eclipse Modeling Framework Project*) que está desarrollado en Java como un *plugin* de Eclipse. Mediante este *framework* se diseñaron dos modelos:

- el modelo de diseño, el cual determina las tareas a ejecutar y el orden en que serán ejecutadas; y
- el modelo de tiempo de ejecución, el cual específica los valores de los parámetros de las tareas y el estado del flujo de trabajo en ejecución.

El *framework* EMF ofrece un soporte para el modelado de flujos de trabajo que permite definir las tareas a llevar y el flujo de ejecución de estas con base en un marco de estrategias, debido a que las tareas pueden ser ejecutadas en serie o paralelo. Esta herramienta ofrece una interfaz grafica donde se modelan los flujos de trabajo. Los componentes de esta herramienta facilitan la definición de las funciones a realizar dentro del flujo de trabajo.

A continuación, se describen los componentes principales de cada uno de los modelos con la finalidad de dar un panorama del uso de dicha herramienta.

Modelo de diseño de las tareas

Está compuesto de un conjunto de componentes importantes, cada uno cuenta con sus respectivas funciones. La Figura 4.10 muestra la estructura que toma un flujo de trabajo; cada componente representa una clase que cuenta con determinados métodos para realizar su función. Estas clases también se relacionan entre sí, lo cual permite que el flujo de trabajo sea diseñado de una manera correcta.

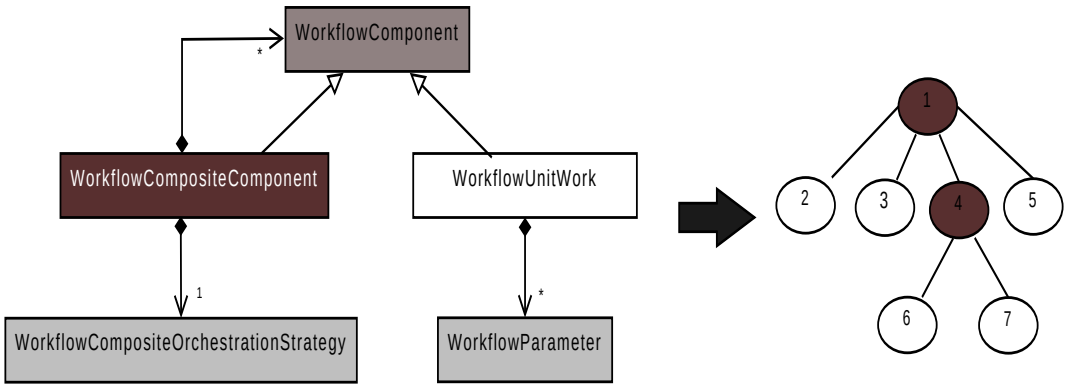

Figura 4.10: Componentes principales para el modelado del flujo de trabajo

Cada uno de los componentes está diseñado para llevar a cabo determinadas funciones:

- `WorkflowComponent`: es el nodo raíz del flujo de trabajo;
- `WorkflowCompositeComponent`: define las sub-tareas padres del flujo de trabajo;
- `WorkflowUnitWork`: se encarga de crear las tareas hijos del flujo de trabajo;
- `WorkflowCompositeOrchestrationStrategy`: permite definir el orden de ejecución de las tareas; y
- `WorkflowParameter`: son los parámetros de entrada y salida de las tareas en tiempo de ejecución.

El árbol de tareas que se muestra a la derecha de la Figura 4.10 representa el orden de las tareas, en tanto que el color de los nodos define la clase que los representa (ver clases de la derecha de la Figura 4.10).

El componente `WorkflowCompositeComponent`, además de determinar los nodos padres del flujo de trabajo, cuenta con un sub-componente que se encarga de la orquestación de las tareas (ver Figura 4.11)

El sub-componente `WorkflowCompositeOrchestrationStrategy` se encarga de precisar la ejecución de las tareas, ya sea de manera secuencial o paralela. La figura 4.11 muestra cómo `WorkflowCompositeOrchestrationStrategy` hereda de los componentes que determinan la secuencia de ejecución. En la parte derecha de esta la figura se ilustra los modos de ejecución secuencial y paralelo de las tareas.

Otro componente principal es `WorkflowComponent` declara el nodo raíz del flujo de trabajo. Sin embargo, también cuenta con el sub-componentes que generan una especie de ciclo para ir pasando de una tarea a otra (ver Figura 4.12).

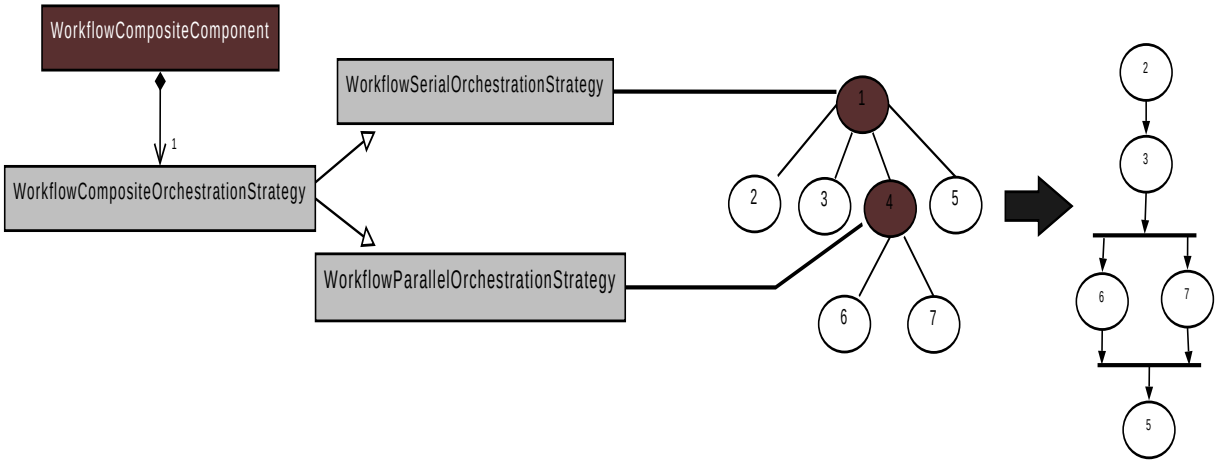

Figura 4.11: Componente `WorkflowCompositeComponent`

Los sub-componentes de `WorkflowComponent` son los siguientes:

- `WorkflowLoopComponentOrchestrationStrategy`: se encarga de definir el ciclo de las tareas y trata de asegurarse de que se cumpla hasta su finalización;
- `kflowComponentOrchestrationStrategy`: determina el modo de orquestación de las tareas;
- `WorkflowExecutionPredicate`: define la ejecución de las tareas; y
- `WorkflowRerunPredicate`: representa la tarea basada en el estado de la última ejecución.

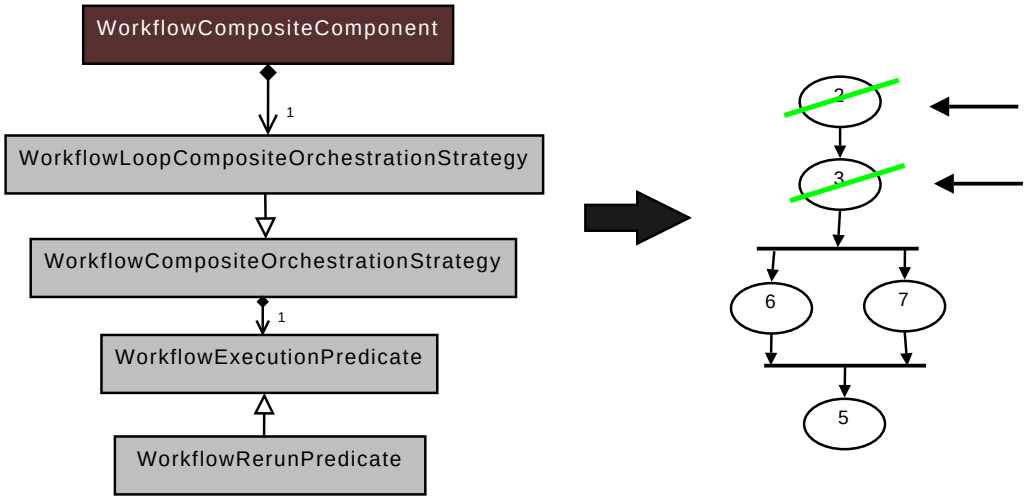

Figura 4.12: Componente `WorkflowComponent`

El lado derecho de la Figura 4.12 ilustra la forma en cómo el ciclo recorre las tareas hasta llegar a la tarea final.

Finalmente otro componente que se contempla en este módulo es el `WorkflowUnitWork`, el cual almacena los parámetros de entrada y salida de cada tarea (ver Figura 4.13).

Los sub-componentes de `WorkflowUnitWork` son los siguientes:

- `WorkflowUnitWork`: representa cada una de las tareas del flujo de trabajo;
- `WorkflowParameter`: almacena los valores de entrada y salida de cada tarea en tiempo de ejecución; y
- `WorkflowParameterConection`: se encarga de conectar un parámetro de salida a un parámetro de entrada.

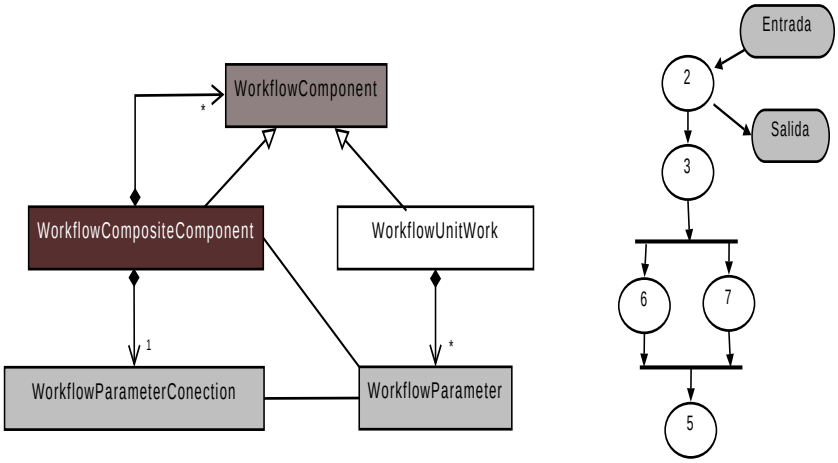

Figura 4.13: Componente `WorkflowUnitWork`

Modelo en tiempo de ejecución

Este modelo se encarga de manejar los valores y los estados del flujo de trabajo durante su ejecución. De esta manera, los valores son evaluados para indicar si las tareas son concluidas o no. Este modelo también es parte componente inicial `WorkflowComponent`(ver Figura 4.14).

Las funciones que realizan los componentes que se engloban en este modelo son las siguientes:

- `WorkflowState`: obtiene el estado de todas las tareas del flujo de trabajo;
- `NoRunning`: indica que la tarea no ha sido ejecutada;
- `DoneState`: notifica que la tarea finalizó su ejecución. Cuando la tarea concluye su ejecución esta debe avaluarse mediante los siguientes sub-componentes:

  1. `SussesState`: informa que la tarea tarea a finalizado con éxito;
  2. `FailedState`: determina que la tarea no concluyó; y
  3. `ErrorState`: notifica que la tarea finalizó pero con excepciones.

- `Running` tarea en ejecución.

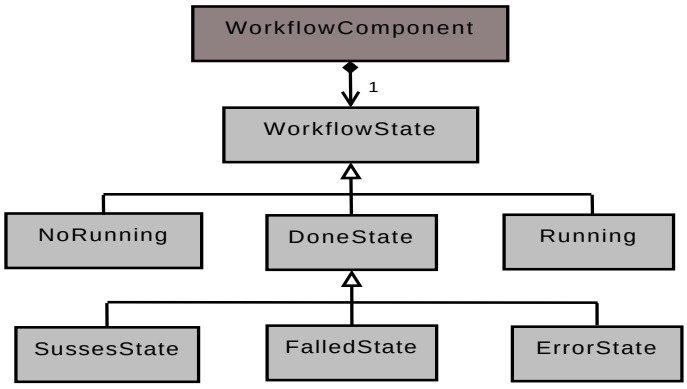

Figura 4.14: Componentes correspondiente al modelo de ejecución

Con la ayuda del *framework* EMF se diseñaron los flujos de trabajo, se describieron cada uno de los componentes con la finalidad de dar un panorama general del comportamiento que el flujo de trabajo va teniendo en base a la construcción de su diseño.

4.4.4 Comunicación entre cliente/servidor y entre áreas

Este componente se encarga de la comunicación entre la aplicación cliente W-RBACSoft y el servidor W-RBACSoft, así como de la comunicación entre los servidores de las áreas. La comunicación se realiza a través de dos mecanismos: 1) el de cliente/servidor, que se utiliza para la comunicación entre la aplicación cliente W-RBACSoft y el servidor del área y 2) el de comunicación P2P (*peer to peer*), que se utiliza en la comunicación entre áreas.

La comunicación consiste en tres fases:

1. **Inicio de conexión:** en el caso de la comunicación cliente/servidor, el dispositivo móvil del usuario inicia la conexión con el servidor para realizar su registro en el sistema W-RBACSoft. En el caso de la comunicación P2P, cualquier área puede iniciar la conexión con alguna otra.

2. **Periodo de comunicación:** en el caso de la comunicación cliente/servidor, el usuario siempre está conectado al servidor W-RBACSoft del área, hasta que se cambie a otra área o termine todas sus actividades. Por otro lado, la comunicación P2P se mantiene activa mientras el área que inicio la conexión permanezca en comunicación con el área que este actuando como servidor.

3. **Terminar conexión:** en la comunicación cliente/servidor, el cliente se encarga de cerrar la conexión con el servidor con el que se encuentre conectado. Por otra parte, la comunicación P2P cierra la conexión cuando el área que se comporta como cliente finaliza conexión con el área que se comporta como servidor.

Comunicación cliente/servidor

Las redes tradicionales están compuestas por dispositivos tales como computadoras, switches, enrutadores, los cuales son básicamente dispositivos fijos. Sin embargo, una infraestructura de red moderna, se integra también de dispositivos móviles tales como PDA (*Personal Digital Assistants*), dispositivos móviles, computadoras portátiles, sistemas embebidos (e.g. un sistema de navegación de un automóvil). El concepto más importante para este tipo de dispositivos es la movilidad; al respecto se puede distinguir dos enfoques de la movilidad [García and Rousseau, 1999]:

- el usuario trabaja en un lugar, se desconecta para moverse y vuelve a conectarse para retomar su trabajo en alguna localidad remota; y
- el usuario sigue conectado aun si se desplaza de un lugar a otro.

En la implementación de este mecanismo, se tomó el segundo enfoque de la movilidad, i.e., el usuario se mantiene conectado mientras se mueve. La Figura 4.15 ilustra las tres fases de la comunicación cliente/servidor, así como la interacción entre la aplicación cliente W-RBACSoft y el servidor W-RBACSoft dentro de una área.

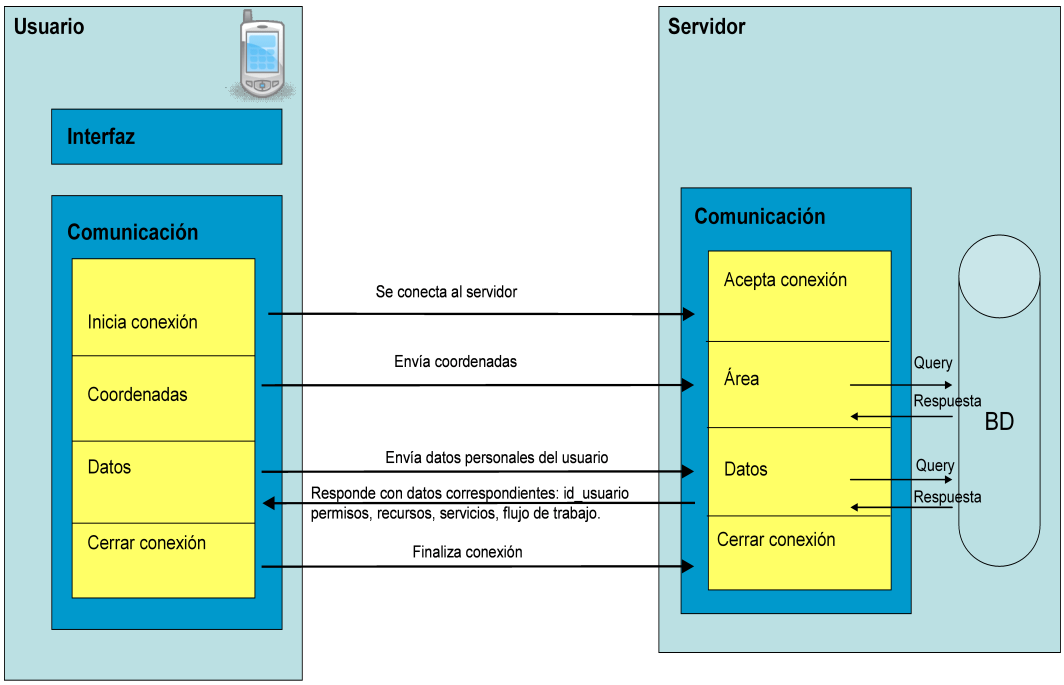

Figura 4.15: Comunicación entre cliente y servidor W-RBACSoft

La comunicación entre el cliente W-RBACSoft (dispositivo móvil) y el servidor W-RBACSoft (PC) se realiza por medio del uso de sockets. En Java un socket es un punto de conexión por el cual un proceso puede emitir o recibir información [M. Hughes, 1999].

En el modelo de sockets de flujo de datos, se cuenta con un objeto de tipo Socket en el cliente y en el servidor un objeto Socket asociado a un ServerSocket. La entrada y

salida de datos se realiza a través de objetos InputStream y OutputStream asociados a los Sockets.

El servidor W-RBACSoft cuenta con un mecanismo capaz de atender a varios clientes al mismo tiempo. Este mecanismo utiliza la clase Thread de java, un thread o hilo es un proceso que se ejecuta en un momento determinado en el sistema. Sin embargo, si se habla de que el servidor debe atender a varios clientes al mismo tiempo basta con entender que en Java, 2 o más procesos pueden ejecutarse al mismo tiempo dentro de una misma aplicación y para ello son necesarios los Threads o hilos.

El procedimiento 3 "Servidor W-RBACSoft" describe el mecanismo que se encarga de atender a los clientes que se conectan al servidor. Cada vez que un cliente se conecta al servidor, crea un hilo para atenderlo, de esta manerá el servidor puede atender a varios clientes a la vez.

Algoritmo 3 Servidor W-RBACSoft

Entrada: $idCliente_{i...n}$: aplicación cliente (i) que solicita la conexión con el servidor

1: i=0
2: crea un *socket* de comunicación por el puerto 2500
3: **mientras** verdadero **hacer**
4: espera conexión
5: crear hilo para atender las peticiones de la aplicación cliente i (cf. procedimiento 4)
6: $i = i + 1$
7: **fin mientras**

El procedimiento 3 "Servidor W-RBACSoft" se encarga de establecer la comunicación con la aplicación cliente W-RBACSoft, los pasos que lleva a cabo este procedimiento se describen a continuación:

- la línea 2 crea un *socket* de tipo servidor, el cual tiene como parámetro el puerto en el que a va estar escuchando las peticiones de la aplicación cliente i. La estructura de esta instrucción es la siguiente:

  ServerSocket ss = new ServerSocket(2500);

- la línea 3 es un ciclo condicional, el cual engloba las instrucciones de las líneas 3-6 estas se encargan de crear nuevo *socket* cada vez que una aplicación cliente solicita conexión.

- la instrucción de la línea 4 está a la espera de conexiones entrantes. Cuando se establece una conexión con la aplición cliente se devuelve una instancia de tipo *socket* con la que se llevará a cabo la comunicación. En este caso s es el *socket* que mantendrá la comunicación con la aplicación cliente i.

  Socket s = ss.accept();

- después de obtener el objeto s que mantendrá la comunicación, la línea 5 crear un hilo que se encarga de atender las peticiones de la aplicación cliente i. Se hace

instancia a la clase `ServidorUnParlante` y se le envía como parametro el *socket s* de comunicación. Está clase será descrita en el procedimiento .

new ServidorUnParlante(s);

- la línea 6 solo determina el número de conexiones que el servidor atiende.

La clase `ServidorUnParlante` es la que atiende las peticiones de la aplicación cliente, en el siguiente procedimiento se describe la funcionalidad de esta clase.

Algoritmo 4 ServidorUnParlante

Entrada: *socket s* es el socket con el que se lleva la comunicación con la aplicación cliente, *tipopeticion* es la variable que se encarga de definir que tipo de petición que solicita la aplicación cliente

```
1:  s = socket;
2:  entrada = s.getInputStream
3:  salida = s.getOutputStream
4:  mientras verdadero hacer
5:     recibe las peticiones de la aplicación cliente
6:     si entrada.readLine() = "0" entonces
7:        recibe el nombre de usuario y contraseña
8:        realiza una consulta a la base de datos para verificar si el usuario está registrado
9:        solicita ubicación del usuario a la clase ServidorCoord
10:       salida.println(datos correspondientes)
11:    si no, si entrada.readLine() = "1" entonces
12:       recibe los datos para llevar a cabo el registro de un usuario interno
13:       genera nombre de usuario y contraseña
14:       realiza el registro del usuario interno en la base de datos
15:       solicita ubicación del usuario a la clase ServidorCoord
16:       salida.println(datos correspondientes)
17:    si no, si entrada.readLine() = "2" entonces
18:       recibe los datos para llevar a cabo el registro de un usuario externo
19:       realiza el registro del usuario externo en la base de datos
20:       solicita ubicación del usuario a la clase ServidorCoord
21:       salida.println(datos correspondientes)
22:    si no, si entrada.readLine() = "FIN" entonces
23:       s.close
24:    fin si
25: fin mientras
```

Los pasos principales del procedimiento 4 correspondiente a la clase `ServidorUnParlante` los cuales se describen a continuación:

- la línea 1 se encarga de obtener el *socket s* para la comunicación con la aplicación cliente

- las líneas 2 y 3 establecen el flujo de entrada y de salida de datos
- la línea 4 es un ciclo condicional, el cual esta en espera de la recepción de las peticiones de la aplicación cliente. Este ciclo engloba las instrucciones de las líneas 5-24
- la línea 5 está en espera de las peticiones que realiza la aplicación cliente, el método que se utiliza es el *readLine()*
- las líneas 6, 11, 17 y 22 se encargan de tratar la información que reciben de la aplicación cliente, la línea 7 lleva a cabo un proceso de autentificación del usuario. El servidor se encarga de verificar si el usuario ya esta registrado en la base de datos (línea 8).
- en caso de que la petición de la aplicación cliente sea la línea 11, se debe realizar el registro del usuario en la base de dato, en esta línea se trata a un usuario interno por lo tanto, el servidor le genera su nombre de usuario y contraseña para que la siguiente ves que acceda al lo haga por la petición de autentificación.
- la línea 17 se encarga de realizar el registro de los usuarios externos a la base de datos
- las líneas 9, 15 y 20 se encargan hacer una instancia a la clase *ServidoorCoord* para obtener la ubicación del usuario y así determinar los permisos y flujo de trabajo que le corresponde al usuario.
- finalmente la línea 22 cierra la conexión entre la aplicación cliente y el servidor.

  Cada una de las peticiones de las líneas 7, 11, 17 y 22 son tratadas por el servidor de manera separada con la finalidad de que responda dicha petición de una manera adecuada.

En el procedimiento 5 "Cliente W-RBACSoft" se observan los pasos necesarios para ejecutar la aplicación cliente en el dispositivo móvil N95. El objetivo de este procedimiento es describir como inicia la conexión la aplicación cliente con el servidor W-RBACSoft.

Algoritmo 5 Cliente W-RBACSoft

Entrada: Sea *recursos[]* los recursos disponibles y *actividades[]* las actividades correspondientes del flujo de trabajo del usuario

1: *url*="socket://IP:2500"
2: inicia la conexión con el servidor (*url*)
3: se establece el flujo de datos de entrada
4: se establece el flujo de datos de salida
5: se crea un objeto de la clase *TareaEnviar* para invocar al método *enviar()*
6: **mientras** verdadero **hacer**
7: espera datos del servidor
8: **fin mientras**

Las instrucciones llevadas a cabo por el cliente W-RBACSof para iniciar la conexión con el servidor, serán descritas acontinuación en base al procedimiento anterior.

- la línea 2 se encarga de iniciar la conexión con el servidor correspondiente a la *url*. La estructura de esta instrucción es la siguiente:

  sock = (SocketConnection)Connector.open(url);

- en las líneas 3 y 4 se establecen los flujos de datos de entrada y salida, las instrucciones correspondientes son:

  entrada = sock.openInputStream();

  salida = sock.openOutputStream();

- la instrucción de la línea 5 se encarga de crear un objeto del tipo de la clase *TareaEnviar*. Este objeto es un hilo que se encarga de controlar el envío de los mensajes al servidor.

  tEnvio = new TareaEnviar();

  tEnvio.start();

  tEnvio.enviar(mensaje);

- la línea 6 es un ciclo condicional, el cual se encarga estar en espera de los datos que envía el servidor.

  Cabe mencionar que cada vez que la aplicación cliente realiza una solicitud al servidor esta se lleva a cabo mediante la invocación del método *enviar()*.

La aplicación cliente W-RBACSoft además de enviar sus peticiones al servidor W-RBACSoft para obtener sus recursos y su flujo de trabajo, también se encarga de enviar constantemente al servidor sus coordenadas de ubicación. Estas coordenadas son enviadas cada 3 segundos de tal manera que se simule la movilidad del usuario. Por lo tanto, el tratamiento de la recepción de coordenadas en el servidor se hizo mediante un puerto distinto con la finalidad de que está información sea tratada de manera independiente.

Los siguientes procedimientos describen el comportamiento del servidor W-RBACSoft y la aplicación cliente W-RBACSoft en base a la recepción y envío de coordenadas.

El servidor W-RBACSoft establece un socket con el puerto 2502 para la recepción de las coordenadas de la aplicación cliente. Las coordenadas que reciba el servidor le permitirá determinar la ubicación del usuario.

Algoritmo 6 Recepción de coordenadas

Entrada: $cliente_{i...n}$: aplicación cliente (i) que solicitan la conexión con el servidor

1: crea un *socket* de comunicación por el puerto 2502
2: **mientras** verdadero **hacer**
3: espera conexión
4: crear hilo para tratar las coordenadas que envía la aplicación cliente (cf. procedimiento 7)
5: **fin mientras**

Algoritmo 7 ServidorCoord

Entrada: *coordenadas* variable que almacena las coordenadas de ubicación del usuario

```
1: entrada = s.getInputStream
2: mientras verdadero hacer
3:    recibe coordenadas(X,Y)
4:    si rango_InicialArea ≥ coordenadas(X,Y) ≤ rango_FinalArea entonces
5:       se determina el área
6:       retorna el área a la clase ServidorUnParlante
7:    fin si
8: fin mientras
```

El procedimiento 6 "Recepción de coordenadas" se encarga de atender la conexión con la aplicación cliente por el puerto 2502 (línea 1). En seguida, se crea un hilo del tipo de la clase *ServidorCoord* que se encarga de dar tratamiento a las coordenadas recibidas (línea 4). El procedimiento 7 "ServidorCoord", establece el flujo de datos de entrada (línea 1), el cual obtendrá las coordenadas del usuario para compararlas con los rangos de ubicación de las áreas, de este modo es como se determina el lugar en el que se encuentra el usuario (líneas 3, 4 y 5). Finalmente se devuelve el área a la clase *ServidorUnParlante* (línea 6).

La aplicación cliente se encarga de enviar las coordenadas de ubicación del usuario al servidor, el siguiente procedimiento 8 "Envío de coordenadas" describe los pasos que realiza cliente W-RBACSoft.

Algoritmo 8 Envío de coordenadas

Salida: *coordenadas* correspondientes a la ubicación del usuario

```
1: url="socket://IP:2502"
2: inicia la conexión con el servidor (url)
3: se establece el flujo de datos de salida
4: se crea un objeto de la clase TareaEnviar2 para invocar al método enviar2()
5: mientras verdadero hacer
6:    esperar 3 segundos
7:    genera números aleatorios (X, Y)
8:    envía coordenadas(X,Y)
9: fin mientras
```

El procedimiento 8 "Envío de coordenadas" realiza los siguientes pasos para enviar las coordenadas al servidor.

- la línea 2 se encarga de iniciar la conexión con el servidor correspondiente de la *url*
- en las línea 3 establece el flujo de datos de salida, ya que solo se encarga de enviar las coordenadas
- la instrucción de la línea 4 se encarga de crear un objeto del tipo de la clase *TareaEnviar2*. Para controlar el envío de los mensajes.

- la línea 6 es un ciclo condicional que engloba varias instrucciones (líneas 6-8) las cuales se encargan de generar números aleatorios cada 3 segundos y con el método *enviar2*, envía las coordenadas al servidor W-RBACSoft.

Comunicación P2P entre las áreas

Para implementar la el mecanismo de la comunicación entre áreas, se utilizó la arquitectura P2P pura o descentralizada. Actualmente este tipo de arquitectura se refiere a la comunicación de dos aplicaciones sin requerir de un servidor intermediario. Una aplicación con está arquitectura puede funcionar como servidor y cliente a la vez, i.e., puede realizar o responder peticiones [Siu and Sai, 2001].

La Figura 4.16 ilustra la comunicación P2P entre áreas, así como la forma en que tiene interacción la aplicación servidor W-RBACSoft de una área con la aplicación servidor W-RBACSoft de otra área.

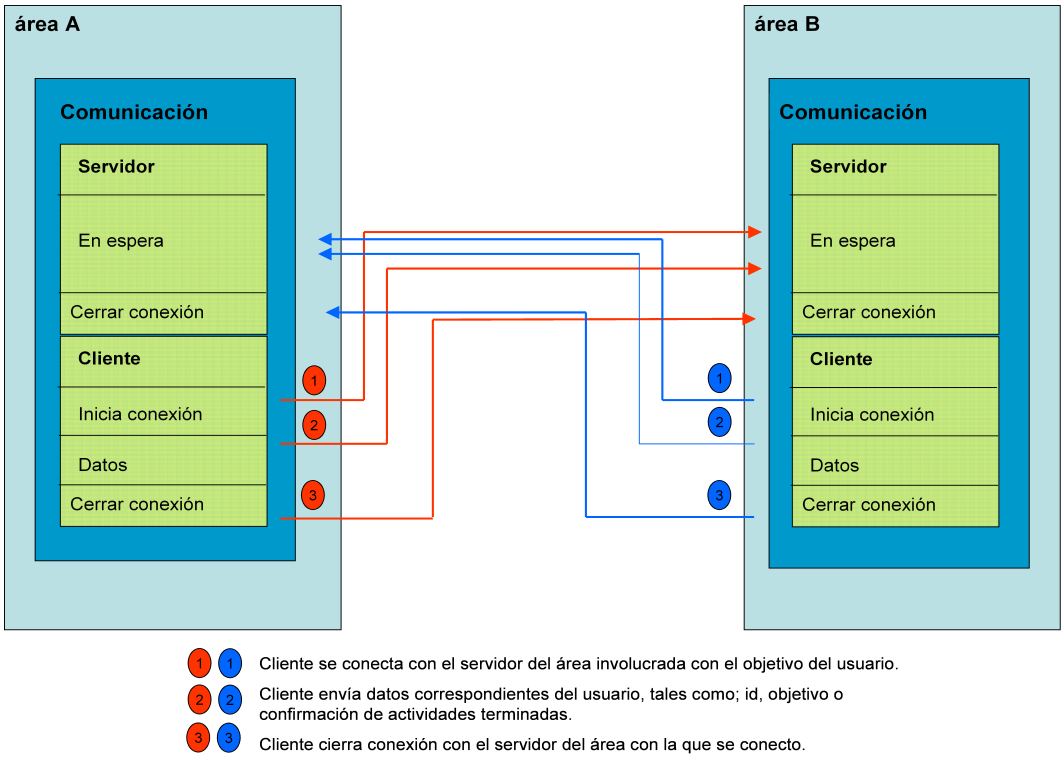

Figura 4.16: Comunicación cliente W-RBACSoft y servidor W-RBACSoft

Como se muestra en la Figura 4.16 ambas aplicaciones de cada área actúan como servidor y como cliente. La parte que actúa como servidor siempre se encuentra activa en espera de que alguien se conecte a ella. Por otro lado, la parte del cliente es la encargada de iniciar la conexión.

La comunicación entre áreas se da cuando el objetivo del usuario las involucra, ya que deben comunicarse entre sí para enviarse información con respecto al usuario que tendrá acceso a ellas. Los siguientes procedimientos describen la forma de como las áreas se comunican. Cada área cuenta con su propio servidor W-RBACSoft, la comunicación entre

áreas se determinó que se realizará por un puerto independiente, el puerto que recibirá la petición de las áreaCliente es el 2503. Los procedimientos descritos a continuación explican la el comportamiento de las áreas con respecto a su comunicación.

Algoritmo 9 Servidor P2P

Entrada: *clienteArea*$_{i...n}$: corresponde al área cliente que solicita la conexión el área que actua como servidor, *objetivo* almacena el objetivo del usuario

1: i=0
2: crea un *socket* de comunicación por el puerto 2503
3: se define áreas involucradas en el objetivo de usuario
4: **mientras** verdadero **hacer**
5: en espera conexión de las áreas involucradas
6: crear hilo para tratar a cada una de las áreas que inicien conexión (cf. procedimiento 10)
7: *i=i+1*
8: **fin mientras**

Algoritmo 10 áreaServidor

Entrada: *datoscliente* almacena los datos del cliente que envía el áreaCliente *i*

1: establece el flujo de datos de entrada
2: establece el flujo de datos salida
3: **mientras** verdadero **hacer**
4: recibe las los datos que envía el áreaCliente
5: lleva a acabo una consulta a la base de datos para hacer el registro del usuario o modificar el estado del usuario
6: **fin mientras**
7: cierra conexión con áreaCliente

Los procedimientos 9 "Servidor P2P" y 10 "áreaServidor" interactúan entre sí para establecer la comunicación con áreaCliente (procedimiento 11). El primer procedimiento se encarga de crear el socket y establecer la comunicación con el áreaCliente, mientras que el segundo procedimiento se encarga de atender las peticiones del áreaCliente. A continuación se describen estos procedimientos.

- en la línea 1, crea el socket de comunicación y determina el puerto donde estará escuchando las peticiones del áreaCliente.
- la línea 2 realiza una consulta a la base de datos para obtener el objetivo del usuario y así determinar que áreas son las involucradas
- la línea 3 es un ciclo condicional el cual engloba las líneas 5-7 que se encargan de establecer la comunicación con el áreaCliente y se genera un hilo el cual hace instancia a la clase *áreaServidor*. El procedimiento 10 "áreaServidor" establece los flujos de datos de entrada y salida para la comunicación con el áreaCliente.

La parte del área que actúa como cliente, se ocupa de iniciar las conexiones con las áreas que estén involucradas en el objetivo del usuario para enviarles información correspondiente del usuario. Cabe mencionar que la comunicación entre áreas se da de manera jerárquica. Las áreas padres se encargan de crear comunicación con las áreas hijo si así se requiere.

El procedimiento 11 "áreaCliente" describe el conjunto de instrucciones que realiza para comunicarse con las áreas correspondientes.

Algoritmo 11 áreaCliente

Entrada: *socket s es el socket con el que se lleva la comunicación con la aplicacóon cliente*

1: *url[]* = "socket://IP:2503"
2: realiza una consulta a la base de datos para obtener las IPs de sus áreas hijo involucradas y las almacena en el arreglo *url[]*
3: **para** *i=0* **a** *url[].size* **hacer**
4: inicia la conexión con la/las áreaServidor (*url[i]*)
5: crea el flujo de datos de entrada
6: crea el flujo de datos de salida
7: se crea un objeto de la clase *TareaEnviar3* para invocar al método *enviar3()*
8: se crea un objeto de la clase *RecibirDatos* para invocar al método *recibir3()*
9: *i=i+1*
10: **fin para**
11: **para** *i=0* **a** *url[].size* **hacer**
12: cierre conexión *i*
13: **fin para**

La descripción del procedimiento 11 "áreaCliente" se explica a continuación:

- en la línea 2 realiza una consulta a las base de datos para obtener la dirección IP de cada área involucrada y estas se guardan en un arreglo llamado *url*.
- la línea 3 se encarga de crear un ciclo condicional del tamaño del arreglo *url* para iniciar la conexión con las áreas. Se establecen los flujos de datos de entrada y salida (línea 5 y 6). Se crea un objeto del tipo de la clase *TareaEnviar3* para enviarár datos a las áreas (línea 7). También se crea un objeto del tipo de la clase *RecibirDatos* para la recepción de los datos de entrada (línea 8).
- Finalmente se crea un ciclo condicional del tamaño del arreglo *url* para cerrar las conexiones con las áreas.

4.5 Pruebas y resultados

Las pruebas referentes que se hicieron al sistema W-RABCSoft se enfocaron a dos tipos de usuarios, internos y externos. Se determinaron IP's y puertos estáticos para cada una de las áreas involucradas tales como: 1) la `Recepción` y 2) el `Departamento de Computación`, áreas correspondientes a una institución educativa.

4.5.1 Usuario interno

Este tipo de usuario se encarga de obtener los permisos a los recursos del área a la que pertenece.

Para autorizar el acceso a la institución el administrador de la `recepción` lleva a cabo un proceso de autentificación en el cual el usuario debe presentar información que respalde que es miembro de la institución. Una vez autorizada la entrada del usuario el administrador le proporciona un dispositivo móvil e inicializa la aplicación cliente W-RBACSoft (ver Figura 4.17).

Figura 4.17: Interfaz del sistema W-RBACSoft

Cuando el usuario inicia la aplicación cliente W-RBACsoft en el dispositivo móvil N95, se inicia la conexión con el servidor W-RBACSoft del área (ver Figura 4.18). En ese momento el usuario inicia la interacción con el sistema.

Figura 4.18: Aplicación cliente W-RBACSoft inicia conexión con el servidor W-RBACSoft

Registro de usuario interno

Cuando el usuario interno ingresa al sistema W-RBACSoft por primera vez, el sistema establece que el usuario debe ser registrado para almacenar su información en la base de

datos del área correspondiente. Para realizar el registro del usuario, se debe interactuar con la aplicación cliente. Primeramente, se selecciona el perfil del usuario para que muestre la interfaz de registro correspondiente (ver Figura 4.19).

Figura 4.19: Registro de un usuario interno

Después de seleccionar el perfil, la aplicación cliente W-RBACSoft muestra la interfaz de registro, a través de la cual el usuario debe ingresar sus datos correspondientes tal como se ilustra en la Figura 4.20.

Figura 4.20: Registro del perfil empleado

Al termino del registro, el usuario debe seleccionar el botón “enviar” (ver Figura

4.21). En respuesta la aplicación cliente envía la información del usuario al servidor W-RBACSoft del área. En esté caso la envía al área de `recepción` (ver Figura 4.22).

Figura 4.21: Envío de datos del usuario al servidor W-RBACSoft de la `Recepción`

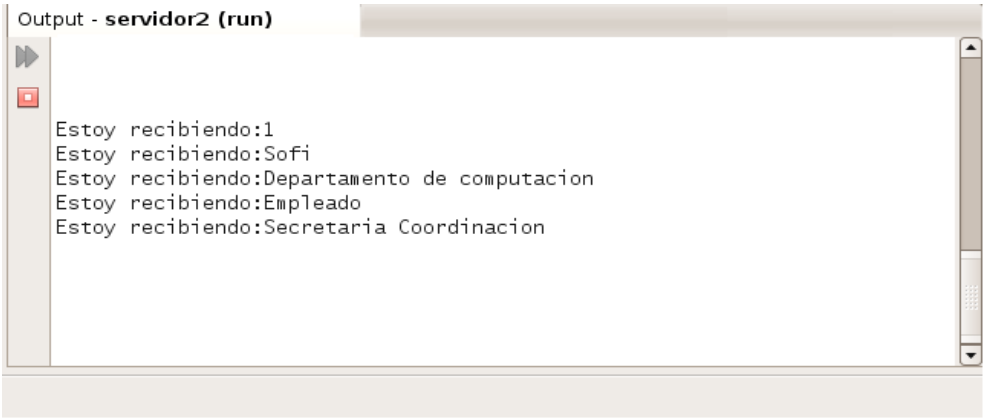

Figura 4.22: Datos recibidos en el servidor W-RBACSoft de la `recepción`

Cuando el servidor W-RBACSoft recibe la información del usuario, la almacena en la base de datos del servidor, genera su nombre de usuario y contraseña, le asigna el rol `empleadoComputacion` y define sus permisos (ver Figura 4.23). También el servidor se encarga de notificar y enviar información del usuario al área `Departamento de Computación` (área destino del usuario) para que esta haga el registro del usuario en su respectiva base de datos.

Cada 3 segundos la aplicación cliente W-RBACSoft envía las coordenadas del usuario al servidor W-RBACSoft de la área `Recepción` para identificar su ubicación. En el momento en que el usuario se acerca al rango de coordenadas del área `Departamento de`

Figura 4.23: Usuario registrado en la base de datos de la `Recepción`

`Computación` la aplicación cliente cierra la conexión con el área *Recepción* y crea una nueva conexión con el servidor W-RBACSoft del área `Departamento de Computación`.

Una vez que el usuario accede al área (`Departamento de Computación`), esta le asigna un nuevo rol `secretariaCoordinacion` y con ello sus nuevos permisos a los recursos.

Cuando el usuario selecciona el botón `Nuevo` la aplicación cliente se encarga de mostrarle la interfaz con los permisos a los recursos que el área (`Departamento de Computación`) le ha asignado. Estos permisos fueron enviados por el servidor W-RBACSoft del área de `Departamento de Computación` (ver Figura 4.24).

Figura 4.24: Permisos del usuario en el área `recepción`

La ubicación del usuario le permite al servidor W-RBACSoft identificar el lugar en el que se encuentra el usuario (e.g en el auditorio, en el laboratorio 1) y determinar si le corresponde un nuevo rol, permisos o actividades. Una vez que el usuario termina con

sus actividades laborales, el usuario accede al área `Recepción` para finalizar la aplicación cliente W-RBACSoft.

Acceso de usuario al sistema W-RBACSoft

Una vez que el usuario ya ha sido registrado en el sistema W-RBACSoft se le proporciona un identificador de usuario y su contraseña. Estos datos le permitirán la entrada al sistema en sus posteriores accesos a la organización (ver Figura 4.25).

Figura 4.25: Entrada del usuario al sistema W-RBACSoft

Suponga que el usuario ingresa su identificador de usuario y contraseña al sistema, está información la obtiene la aplicación cliente y la envía al servidor W-RBACSoft. El servidor recibe la información y realiza una consulta a la base de datos del área para verificar si el identificador y contraseña del usuario pertenece a alguno de los usuarios ya registrados en la base de datos, si estos datos coinciden el servidor autoriza el acceso del usuario y le proporciona sus permisos correspondientes (ver Figura 4.26)

Figura 4.26: Servidor W-RBACSoft verifica si el usuario esta registrado

4.5.2 Usuario externo

Este tipo de usuario siempre deberá llevar a cabo el proceso de registro, ya que su estancia a la institución es temporal. Al igual que el usuario interno lleva a cabo una serie de selección de opciones en base a su persona para realizar su registro (ver Figura 4.27).

Figura 4.27: Usuario externo

Al seleccionar el perfil del usuario externo en la aplicación cliente W-RBACSoft le muestra la interfaz correspondiente a su perfil para que lleve a cabo su registro. En este caso se hizo la prueba con el perfil `Investigador` y se procedío a realizar el registro del usuario (ver Figura 4.28).

Después de ingresar y seleccionar los datos personales del usuario, fueron enviados al servidor W-RBACSoft el cual lleva a cabo el registro del usuario en la base datos y el tratamiento de la información. En los datos que proporciona el usuario se encuentra el objetivo por el cual requiere acceso a la institución. Por lo tanto, el servidor asigna sus permisos y su flujo de traabajo. También se encarga de comunicarle al área destino del objetivo del usuario (ver Figura 4.29).

Como se puede ver en la figura anterior el perfil `Investigador` tiene como objetivo `dar conferencia` y su área destino es `Departamento de Computación`. Para este tipo de usuario el sistema no solo se encarga de enviarle los permisos a los recursos del área sino también su flujo de trabajo. Está información es enviada por el servidor W-RBACSoft y procesada por la aplicación cliente W-RBACSoft para mostrarla en la interfaz del usuario de una manera amigable (ver Figura 4.30). Sin embargo, el servidor W-RBACSoft del área `Recepción` se encarga de informarle al área `Departamento de computación` que el investigador invitado accederaá a ella para dar la conferencia programada.

Figura 4.28: Datos de registro del usuario externo

Figura 4.29: Datos de usuario externo son enviados al servidor W-RBACSoft

Cuando las coordenadas del usuario se acercan al rango de coordenadas del área `Departamento de computación`, la aplicación cliente W-RBACSoft termina la conexión con el área `Recepción` e inicializa conexión con el área `Departamento de Computación`. Esta área como ya tiene conocimiento de la llegada del usuario y su respectivo objetivo le asigna un nuevo rol al usuario (`investigador_InvitadoComputacion`) y le determina sus

permisos a los recursos, asś como su flujo de trabajo a realizar. Como se vío en la figura 4.30 el usuario podra tener interacción con la aplicación cliente y podrá elegir si desea ver sus recursos o su flujo de trabajo (ver Figura 4.31).

Figura 4.30: Menú de usuario externo

Figura 4.31: Interfaz de recursos y flujo de trabajo del usuario

Cabe mencionar que si el usuario tuviera acceso a otra área, se activa la opción `Nuevo` para mostrarle los nuevos permisos y flujo de trabajo asignados por la nueva área. También la comunicación entre las áreas se mantiene ya que deben mantenerse informadas del estado del usuario.

Capítulo 5

Conclusiones y trabajo futuro

El presente capítulo está estructurado en tres partes. Primeramente se recapitula la problemática abordada en esta tesis de maestría (cf. sección 5.1). Enseguida, se presenta una síntesis de las contribuciones de este trabajo de investigación y desarrollo (cf. sección 5.2). Finalmente, se presenta algunas limitaciones y posibles extensiones futuras del sistema W-RBACSoft (cf. sección 5.3).

5.1 Resumen de la problemática

Este trabajo de tesis se enfoca en tres campos de investigación de las ciencias computacionales: 1) el cómputo ubicuo, 2) el control de acceso basado en roles, y 3) los flujos de trabajo (*workflows*). El objetivo del cómputo ubicuo es crear entornos físicos inteligentes mediante el uso de sistemas computacionales que estén disponibles, pero que sean invisibles al usuario. El control de acceso basado en roles facilita la atribución de permisos y restricciones a los usuarios, ya que permite describir un grupo de usuarios que, bajo un conjunto de roles, puede realizar operaciones sobre un conjunto de objetos. Finalmente, los flujos de trabajo se encargan de automatizar los procesos que engloba una organización.

En la actualidad, las organizaciones cuentan con sistemas de información que están integrados por un conjunto de componentes, que interactúan entre sí, con el fin de apoyar sus actividades y controlar sus procesos. Estos sistemas contemplan tres funciones básicas referentes a la entrada/salida, al procesamiento y al almacenamiento de la información. La agrupación de estas funciones conforman un proceso. Generalmente, en una organización de cualquier índole se lleva a cabo una gran cantidad de procesos.

La tecnología de flujos de trabajo se encarga precisamente de proporcionar un entorno automatizado para apoyar de manera efectiva la realización de los procesos de una organización. Esta tecnología abarca desde la especificación formal de procesos hasta la monitorización y ejecución de los mismos. Sin embargo, la tecnología de flujos de trabajo no contempla algunos aspectos fundamentales que contribuirían a lograr un mejor desempeño de la organización como: a) la implantación de un soporte distribuido de manera estructural que facilite la administración de recursos e b) la incorporación de un mecanismo que controle el acceso de los usuarios nómadas a los recursos. Por lo tanto, la

tecnología de flujos de trabajo para entornos organizacionales deja un campo abierto a la investigación, que da origen a la integración de dichos aspectos.

5.2 Conclusiones

Una organización está compuesta de varias áreas, las cuales manejan grandes cantidades de información, se enfocan en la relación de tareas específicas y cuentan con múltiples recursos heterogéneos. En consecuencia, resulta evidente la necesidad de contar con una herramienta que se encargue de administrar, de manera distribuida, todos los elementos que conforman la organización como: 1) la información contextual de los usuarios en cada área, 2) las actividades realizadas en cada proceso y 3) los recursos utilizados por los usuarios para lograr sus objetivos. En respuesta a esta necesidad, se propuso el sistema W-RBACSoft, cuya la contribución principal, a nivel de diseño, concierne una administración basada en áreas de una organización. Esta solución propone una administración distribuida y estructurada de la organización que pretende facilitar las actividades de los trabajadores y visitantes. De esta manera, cada área que compone la organización se encarga de administrar su propia información, sus procesos y sus recursos, con el fin de poder atribuir actividades específicas a los usuarios.

La arquitectura de *software* del sistema W-RBACSoft está compuesta por dos componentes esenciales: 1) el servidor W-RBACSoft y 2) el cliente W-RBACSoft.

El servidor W-RBACSoft se encarga de administrar el control de acceso, los flujos de trabajo, la información contextual y los recursos de su respectiva área, ya que este componente está alojado en cada área de la organización. Sin embargo, el servidor W-RBACSoft asociado a cada área debe ser capaz de comunicarse con los demás, ya que la movilidad de los usuarios y el intercambio de información entre las áreas son inevitables para el logro de los objetivos de los usuarios. Así mismo, el servidor W-RBACSoft contempla la ubicación del usuario, la cual es determinada mediante un simple mecanismo de simulación que consiste en delimitar a las áreas por un rango de coordenadas para determinar el posicionamiento del usuario.

El cliente W-RBACSoft se encarga de representar al usuario y de interactuar con el servidor W-RBACSoft con la finalidad de proporcionar a dicho usuario la información apropiada (e.g., recursos y flujos de trabajo) para que pueda alcanzar sus objetivos. Dicha información es accesible desde el dispositivo móvil del usuario. Las funciones que engloba la aplicación cliente son: 1) transferir las coordenadas de ubicación del usuario al servidor W-RBACSoft correspondiente; 2) enviar la información del usuario al servidor; y 3) recibir de parte del servidor los recursos y flujos de trabajo requeridos por el usuario.

La comunicación entre áreas utiliza la arquitectura P2P en tanto que la comunicación entre el servidor W-RBACSoft de un área y la aplicación cliente W-RBACSoft emplea obviamente la arquitectura cliente/servidor.

El servidor W-RBACSoft está formado por dos componentes principales, que se encargan respectivamente de controlar el acceso de los usuarios nómadas a las áreas autónomas y de definir los flujos de trabajo que guían las actividades de dichos usuarios dentro de tales áreas.

El componente de control de acceso se encarga de determinar los permisos y las restric-

ciones de los usuarios dentro de la organización, específicamente en las áreas en donde se encuentran. Este componente está basado en el modelo RBAC, el cual ofrece una administración segura de la organización, gracias a la asignación de roles a los usuarios. Cada rol activa ciertos permisos, los cuales son definidos como acciones permitidas sobre los recursos (e.g., utilizar una impresora).

El componente de flujos de trabajo se encarga de determinar las actividades que los usuarios deben llevar a cabo dentro de la organización con base en su objetivo. Para especificar estas actividades se utilizó el *framework* EMF disponible en el entorno de desarrollo de Eclipse. Este *framework* está compuesto de dos partes: el modelo de diseño y el modelo de ejecución. Mediante estos modelos se define los flujos de trabajo por cada objetivo del usuario y a su vez la secuencia que estos deben seguir en tiempo de ejecución. El flujo de trabajo otorgado al usuario se vuelve más específico a medida que accede a cada una de las áreas involucradas, ya que sería complicado y laborioso gestionar un flujo de trabajo general.

De acuerdo a la descripción dada de cada componente, se puede apreciar que la administración basada en áreas de una organización ofrece a los usuarios un mayor apoyo en sus actividades y a la organización un mejor control de sus recursos.

5.3 Trabajo futuro

El proceso de desarrollo del sistema W-RBACSoft ha permitido descubrir algunas extensiones posibles, las cuales han sido clasificadas en los siguientes rubros.

Entre las mejoras que se puede realizar a los componentes del sistema W-RBACSoft que han sido implementados se encuentra:

a) La implantación del sistema W-RBACSoft en una red global, e.g., internet, ya que por ahora las pruebas se realizaron en una red local.

b) Las extensiones de algunos de los componentes del servidor W-RBACSoft, así como la incorporación de un componente de seguridad, se describen en los siguientes puntos:

   1. Actualmente, los flujos de trabajo están pre-definidos para cada uno de los objetivos de los usuarios que se contemplaron. Una de las extensiones que podría realizarse a este componente consiste en manejar flujos de trabajo dinámicos. De esta manera, el usuario podría interrumpir sus actividades para llevar a cabo otras que forman parte de un flujo de trabajo distinto; entre tanto, las actividades del primer flujo de trabajo se mantendrían en espera hasta que el usuario las retome nuevamente. Otra extensión posible de este componente se refiere al manejo de excepciones, que proporcionen información al usuario cuando este haya realizado alguna acción incorrecta.
   2. Para calcular la localización del usuario, se creó un mecanismo que simula su ubicación. El cliente genera coordenadas de localización las cuales son enviadas constantemente al servidor para simular la movilidad del usuario en la organización. Sin embargo, el cliente y el servidor W-RBACSoft requieren conocer

las coordenadas del usuario para realizar funciones distintas. Por ejemplo, la aplicación cliente requiere la ubicación del usuario para poder crear la conexión con el servidor del área actual; de igual manera, el servidor define el rol, los permisos y el flujo de trabajo del usuario en función de su localización. Para calcular la ubicación real del usuario dentro de un edificio se propone desarrollar una herramienta que monitore constantemente la movilidad del dispositivo móvil del usuario para determinar su ubicación de manera automática.

3. No fue implementado ningún componente que garantice la seguridad de la información que se transmite entre cliente-servidor y servidor-servidor, pero es un aspecto importante ya que la información que viaja por la red podría ser vista o violada por otro sistema. Por ejemplo, si la información relevante de un usuario (nombre de usuario y contraseña) es obtenida por algun sistema ajeno, esto provocaría que personas no autorizadas accedan a la información y a los recursos de la organización.

c) En la aplicación cliente W-RBACSoft se propone un mecanismo que se encargue de actualizar la información de manera transparente para el usuario, este mecanismo corresponde a:

1. Actualmente, la aplicación cliente W-RBACSoft cuenta con una interfaz de usuario amigable. Sin embargo, puede volverse algo confusa ya que el usuario debe cambiar de vista cada vez que recibe nuevos recursos y flujos de trabajo dentro de la organización. Para resolver esta limitación se propone, incorporar a la aplicación cliente mecanismos que se encarguen de mostrar la información a los usuarios de manera automática. Estos mecanismos deben activarse con base a la movilidad del usuario. El objetivo de está propuesta es que los recursos y flujos de trabajo sean mostrados al usuario sin necesidad de que él los solicite a la aplicación cliente, más bien que la información se actualice y se muestre automáticamente en el dispositivo móvil cada vez que el usuario cambia de ubicación.
2. Otra extención a la interfaz de usuario consiste en mostrar automáticamente al usuario la ruta que debe seguir para dirigirse al área a la que debe acceder.

Publicaciones del autor

[Mendoza et al., 2009] S. Mendoza, V. Gómez, M. Navarrete, D. Decouchant, K. García, G. Olague and J. Rodríguez, Area-based Collaborative Ubiquitous Work within Organizational Environments, In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp. 140-144, IEEE Computer Society, Milan (Italy), September 15-18 2009

Referencias

[A., 2005] A., M. (agosto, 2005). *Agent-Oriented Approach to Ubiquitous Computing*. Springer Berlin / Heidelberg, Computer Science, pp. 30-37.

[Abowd and Mynatt, 2000] Abowd, G. D. and Mynatt, E. D. (2000). *Charting Past, Present, and Future Research in Ubiquitous Computing*. ACM Trans. Comput.-Hum. Interact., Vol. 7, No. 1, pp. 29-58.

[Aizawa et al., 2004] Aizawa, K., Tancharoen, D., Kawasaki, S., and Yamasaki, T. (2004). *Efficient retrieval of life log based on context and content*. CARPE'04: Proceedings of the the 1st ACM workshop on Continuous archival and retrieval of personal experiences, pp.22-31.

[Androutsellis-Theotokis and Spinellis, 2004] Androutsellis-Theotokis, S. and Spinellis, D. (2004). *A Survey of Peer-to-Peer Content Distribution Technologies*. ACM Computing Surveys, Vol. 36, No. 4, pp. 335-371.

[Bardram and Henrik, 2007] Bardram, E. J. and Henrik, B. C. (January - March 2007). *Pervasive Computing Support for Hospitals: An Overview of the Activity-Based Computing Project*. IEEE Computer Society, pp. 51 - 57.

[Becker and Durr, 2005] Becker, C. and Durr, F. (2005). *On location models for ubiquitous computing*. Springer-Verlag, Personal Ubiquitous Comput., Vol. 9, No. 1, pp. 20-31.

[Cook, 1990] Cook, W. R. (1990). *Object-Oriented Programming Versus Abstract Data Types*. Proceeding of the REXWorkshop/School on the Foundations of Object-Oriented Languages (FOOL), 489, Springer Lecture Notes in Computer Science, pp. 151-178.

[Cuppens and Miége, 2003] Cuppens, F. and Miége, A. (August 2003). *Modelling Contexts in the Or-RBAC Model*. In proceedings of the 19th Annual Computer Security Applications Conference (ACSA), pp. 416, IEEE Computer Society, Washintong, DC(USA).

[de Malaga, 2003] de Malaga, U. (2003). *Java a Tope J2ME (Java 2 MicroEdition)*. pp 10-20.

[Dekker et al., 2008] Dekker, M. A. C., Crampton, J., and Etalle, S. (2008). *RBAC administration in distributed systems*. SACMAT '08: Proceedings of the 13th ACM

symposium on Access control models and technologies, Estes Park, CO, USA, pp. 93-102.

[Edelstein, 1994] Edelstein, H. (1994). *Unraveling Client/Server Architecture*. DBMS, Vol 5, No. 7, pp. 5, May.

[Favela et al., 2007] Favela, J., Tentori, M., Castro, L. A., Gonzalez, V. M., Moran, E. B., and Martínez, A. I. (2007). *Activity recognition for context-aware hospital applications: issues and opportunities for the deployment of pervasive networks*. Kluwer Academic Publishers, Hingham, MA, USA, Vol. 12. No. 2-3, pp. 155-171.

[Ferraiolo et al., 2001] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., and Chandramouli, R. (August 2001). *Proposed NIST Standard for Role Based Access Control*. ACM Transactions on Information and System Security, Vol. 4, No. 3, pp. 224 - 274.

[Ferriolo and Kuhn, 1992] Ferriolo, D. F. and Kuhn, R. D. (1992). *Role-Based Access Controls*. 15th National Computer Security Conference, Baltimore MD, pp. 554 - 563.

[Freudenthal et al., 2002] Freudenthal, E., Pesin, T., Port, L., Keenan, E., and Karamcheti, V. (July 2002). *DRBAC: Distributed role-based access control for dynamic coalition environments*. In Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS), pp. 660 - 665, IEEE Computer Society, Vienna (Austria).

[García and Rousseau, 1999] García, J. A. and Rousseau, F. (1999). *Servicios Activos para Abordar Problemas en Ambientes Móviles*. 2do Encuentro Nacional de Cómputo (ENC99). Pachuca, Hidalgo, México.

[Gerónimo and Canseco, 2002] Gerónimo, G. and Canseco, V. (septiembre - diciembre 2002). *Breve Introducción a los Sistemas Colaborativos: Groupware & Workflow*. Universidad Tecnológica de Mixteca - IEC.

[Gosling et al., 2005] Gosling, J., Joy, B., Steele, G., and Bracha, G. (2005). *The Java Language Specification*. Third Edition, ADDISO-WESLEY.

[Gwan et al., 2003] Gwan, H., Yung-Chuan, L., and Bor-Yih, W. (2003). *A New Language to Support Flexible Failure Recovery for Workflow Management Systems*. Springer, CRIWG, pp. 135-150.

[Han et al., 2006] Han, J., Cho, Y., Kim, E., and Chio, J. (May 8 - 11, 2006). *A Ubiquitous Workflows Service Framework*. In Proceedings of ICCSA 2006 International Conference on Computational Science and its Aplications, LNCS 3983 Springer, pp. 30 - 39, Glasgow (UK).

[Ibarra and Mañas, 2003] Ibarra, N. H. and Mañas, A. J. A. (octubre, 2003). *RBAC: Alternativa actual para la realización de Control de Accesos a gran escala*. Actas del Segundo Congreso Iberoamericano de Seguridad Informática CIBSI 03.

[Kalam et al., 2003] Kalam, A., Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A. Saurel, C., and Trouessin, G. (Julio-25-2003). *Organization based access control*. In Proceedings of IEEE 4th International Workshop on Policies for Distributed Systems and Networks (POLICY 2003), pp. 120- 131, Toulouse France.

[Li et al., 2008] Li, P., Cho, Y., and Choi, J. (2008). *CAWET: A Tool of Context-Aware Workflow with Hand-Held Devices*. ASEA '08: Proceedings of the 2008 Advanced Software Engineering and Its Applications, IEEE Computer Society, pp. 194-197.

[M. Hughes, 1999] M. Hughes, M. Shoffner, D. H. M. W. (1999). *Java Network Programming*. 2da (Ed.), pp. 256-261, Editorial Manning.

[Nicodemos et al., 2001] Nicodemos, D., Naranker, D., Emil, L., Nicodemos, D., and D., N. (January 2001). *The Ponder Policy Specification Language*. In International Workshop, Policies for Distributed Systems and Neworks (Policy 2001), Bristol, UK.

[Oh and Park, 2002] Oh, S. and Park, S. (September 4 - 8, 2002). *Taks-role Based Access control: An Improved Access Control Method for Enterprise Environment*. In proceeding of 11th International Conference on Database and Expert Systems Aplications (DEXA), LNCS 1873 Springer, pp. 264 - 273, London (UK).

[Omicini et al.,] Omicini, A., Ricci, A., and Viroli, M. *RBAC for Organisation and Security in an Agent Coordination Infrastructure*. Elsevier Science Publishers B. V., Vol. 128, No. 5, pp. 65-85, YEAR =.

[Ravi et al., 1996] Ravi, S. S., Edward, J. C., Hal, L. F., and Charles, E. Y. (1996). *Role-Based Access Control Models*. IEEE Computer, Vol. 29, No. 2, pp. 38-47.

[Roth, 2006] Roth, J. (2006). *Detecting identifiable areas in mobile environments*. SAC '06: Proceedings of the 2006 ACM symposium on Applied computing, ACM, pp. 986-991.

[Schoder and Fischbach, 2008] Schoder, D. and Fischbach, K. (2008). *RBAC administration in distributed systems*. Communications of the ACM, Vol. 46, No. 2, pp. 27-29,.

[Silberschatz et al., 2002] Silberschatz, A., F. Korth, H., and Sudarshan, S. (2002). *FUNDAMENTOS DE BASES DE DATOS*. cuarta edición, McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U.

[Siu and Sai, 2001] Siu, M. L. and Sai, H. K. (2001). *Interoperability of Peer-To-Peer File Sharing Protocols*. ACM SIGecom Exchanges, Vol. 3, No. 3, pp. 25-33, August.

[Swenson and Kent, 1995] Swenson, D. K. and Kent, I. (1995). *Workflows Technology: Tradeoffs for Bussiness Process Re-engineering*. ACM , COOCS milpitas , CA (USA).

[Tejedor and Jesús, 2006] Tejedor, M. and Jesús, R. (2006). *Domine las Redes P2p (Peer To Peer)*. 1er Edition, Creaciones Copyright.

[Tentori and Favela, 2008] Tentori, M. and Favela, J. (April - June 2008). *Activity-Aware Computing for Healthcare*. IEEE Pervasive Computing, pp. 51 - 57.

[Wainer et al., 2003] Wainer, J., Barthelmess, P., and Kumar, A. (2003). *W-RBAC - A workflow security model incorporating controlled overriding of constraints*. International Journal of Cooperative Information Systems, Vol. 12. No. 4, pp. 455-485.

[Weiser, 1999] Weiser, M. (1999). *The computer for the 21st century*. SIGMOBILE Mob. Comput. Commun. Rev. of the ACM, Vol. 3, No. 3, pp. 3 - 11.

[Weiser, 1993] Weiser, M. (Julio 1993). *Some Computer Science Problems in Ubiquitous Computing*. Communications of the ACM, 36(7):75-84.

[Worsley and Drake, 2002] Worsley, J. and Drake, J. (2002). *Practical PostgreSQL*. Commandprompt, Inc.

[Zur Muehlen, 2004] Zur Muehlen, M. (2004). *Organizational Management in Workflow Applications – Issues and Perspectives*. Kluwer Academic Publishers, Hingham, MA, USA, Vol. 5. No. 3-4, pp. 271-291.