Integrando Sensibilidad al Contexto, Folcksonomías y Ontologías a Sistemas de Administración de Proyectos

Autora: Ing. Elina María Avila Ordóñez

Director: Dr. Gustavo Rossi

Tesis Presentada para obtener el grado de

Magíster en Ingeniería de Software

FACULTAD DE INFORMÁTICA

UNIVERSIDAD NACIONAL DE LA PLATA

La Plata, Junio de 2011

Contenido

Contenido 2

Índice de Imágenes 5

Índice de Ilustraciones 6

Índice de Diagramas 7

Índice de Tablas 8

Dedicatoria 9

A Iván, Julián y Ezequiel 9

Agradecimientos 10

Introducción 11

1.1. Introducción 11

1.2. Objetivo de la Tesis 13

1.2.1. Objetivo General 13

1.2.2. Objetivos Específicos 13

Administración de Proyectos 14

1.3. Introducción 14

1.4. Conceptos 14

1.5. Framework PMI 15

1.6. Sistemas para la Administración de Proyectos 19

1.6.1. Open WorkBench 19

1.6.2. PhProjekt 20

1.6.3. OpenProj 21

1.7. Algunas oportunidades de mejora de los sistemas para Administración de Proyectos 23

Aplicaciones Sensibles al Contexto 24

1.8. Introducción 24

1.9. Inicios 25

1.10. Contexto, Aplicaciones Sensibles al Contexto y Tipos de Información Contextual 25

1.11. Uso del Contexto en aplicaciones 27

1.12. Arquitecturas 29

Ontologías 33

1.13. Concepto 33

1.14. ¿Por qué utilizar ontologías en el software? 33

1.15. Elementos de una ontología 34

1.16. Lenguajes ontológicos 35

1.16.1. Resource Description Framework RDF 36

1.17. Ontology Web Language OWL 37

1.18. Algunas Aplicaciones 38

Folksonomías 40

1.19. Introducción 40

1.20. Concepto 41

1.21. Folksonomías en Internet 41

1.22. Ventajas y Desventajas 43

Aplicando Sensibilidad al Contexto a sistemas para la Administración de Proyectos 45

1.23. Introducción 45

1.24. Integrando Sensibilidad al Contexto 46

1.25. Definiendo los contextos de interés 48

1.26. Captando el contexto de interés 56

1.27. Consiguiendo información de forma automática 59

1.28. Otras acciones a tomar 60

1.29. Usando el contexto para asistir a la toma de decisiones 65

1.30. Usando la información del pasado 70

Métodos para obtener información de interés 71

1.31. Introducción 71

1.32. Usando folcksonomías 71

1.33. Usando ontologías 78

Aplicando el modelo de clases como extensión de sistemas 91

1.1. Introducción 91

1.2. Fragmentos esenciales 91

1.2.1. Cambios Contextuales 92

1.2.2. Etiquetado Folcksonómico 95

1.2.3. Ontologías 97

El Prototipo 98

1.1. Introducción 98

1.2. Menú 98

1.3. Opción: Probar Cambio Contextual 99

1.4. Opción: Probar Etiquetado con Folcksonomías 102

1.5. Opción: Probar Etiquetado con Ontologías 105

2. Conclusiones y Trabajos Futuros 112

2.1. Conclusiones 112

2.2. Trabajos Futuros 113

Referencias 114

Anexo A Ontología de Prueba 116

Anexo B Herramientas para el manejo de Ontologías 126

Anexo C Instancias 127

Índice de Imágenes

Imagen 1: Algunas pantallas de Open WorkBench 20

Imagen 2: Algunas pantallas de PHProjekt 21

Imagen 3: Pantalla de OpenProj..... 22

Imagen 4: Página Del.icio.us 42

Imagen 5: Menú principal del prototipo 99

Imagen 6: Selección de un estado contextual para un componente..... 101

Imagen 7: Opciones del submenú de pruebas de folcksonomías..... 102

Imagen 8: Selección de una etiqueta folcksonómica para un ProductoComponente..... 103

Imagen 9: Ubicación de un ProductoComponente 104

Imagen 10: Información Relacionada obtenida con el uso de Folcksonomías 105

Imagen 11: Arbol de clases ontológicas..... 106

Imagen 12: Opciones para la prueba de Ontologías..... 106

Imagen 13: Selección de la propiedad a definir par aun ProductoComponente..... 108

Imagen 14: Selección de individuos para relacionar..... 108

Imagen 15: Selección del individuo ontológico..... 110

Imagen 16: Información relacionada obtenida con el uso de Ontologías 110

Índice de Ilustraciones

Ilustración 1: Actividades de un Proyecto 15

Ilustración 2: Áreas de conocimiento. Fuente: PMI 16

Ilustración 3: Situación de dos personas 26

Ilustración 4: Reutilización de una ontología por varios sistemas 34

Ilustración 5: Elementos de una Ontología 35

Ilustración 6: Representación de una Tripleta 36

Ilustración 7: Uso de URIs 37

Ilustración 8: Relaciones de la Administración de Proyectos 46

Ilustración 9: Fuentes de las Necesidades de un Proyecto 47

Ilustración 10: Estado Contextual 50

Ilustración 11: Efecto de un cambio contextual 51

Ilustración 12: Funcionamiento de Sensores 56

Ilustración 13: Un observador de los cambios contextuales 57

Ilustración 14: Datos relacionados a un riesgo 69

Ilustración 15: Etiquetas consignadas a documentos 73

Ilustración 16: Información Relacionada por medio de Etiquetas 75

Ilustración 17: Conexiones a través de etiquetas Folcksonómicas 77

Ilustración 18: Significado de un término según el dominio 78

Ilustración 19: Entrega de información sobre un ProductoComponente 80

Ilustración 20: Proceso para determinar el concepto ontológico adecuado 81

Ilustración 21: Esquema de crecimiento para la ontología a aplicar 85

Ilustración 22: Propiedades de la Clase miDocumento 87

Ilustración 23: Inserción de un individuo de miDocumento 88

Índice de Diagramas

Diagrama 1: Clases Proyecto y Componente 48

Diagrama 2: Especialización de la clase Componente 49

Diagrama 3: Definición de Estados Contextuales y Acciones a tomar 52

Diagrama 4: Instancias creadas ante un cambio contextual 54

Diagrama 5: Observadores 58

Diagrama 6: Actualización de un estado contextual de un Componente 59

Diagrama 7: Clase Agendar 61

Diagrama 8: Clase Publicar 62

Diagrama 9: Instancias para la publicación 63

Diagrama 10: Clase Ejecutar 64

Diagrama 11: Incorporando mayor información 65

Diagrama 12: Instancias de un Proyecto 67

Diagrama 13: Clase ProveerInformación 68

Diagrama 14: Clase ProductoComponente 72

Diagrama 15: Clases para el empleo de Folcksonomías 74

Diagrama 16: Relación Lectura 76

Diagrama 17: Clase Ontología 86

Diagrama 18: Clases Esenciales para extender software 92

Diagrama 19: Aware-Model Componente 93

Diagrama 20: Herencia Multiple 94

Diagrama 21: Aware- Model Componente 95

Diagrama 22: Diseño para Folcksonomías 96

Diagrama 23: Extendiendo Folcksonomías 96

Índice de Tablas

Tabla 1: Ejemplo de aplicación de Sensibilidad al Contexto 26

Tabla 2: Aware Objects 30

Tabla 3: Aware Models..... 31

Tabla 4: Propiedades definidas para unos documentos 83

Dedicatoria

A Iván, Julián y Ezequiel

Agradecimientos

A todos quienes hacen posible la Maestría en Ingeniería de Software, por su cordialidad y amistad en especial a Gustavo Rossi, quien acompañó desde la distancia el desarrollo de este trabajo.

A mis padres, por siempre ser fuente insaciable de apoyo.

Introducción

1.1. Introducción

La comunicación y la organización son dos factores claves para la finalización de un proyecto en tiempo y forma. Aún cuando un proyecto posea un objetivo definido, recursos suficientes y calificados, tiempo y tecnología acorde a las necesidades, podría enfrentarse a problemas si no se establecen mecanismos de difusión y coordinación.

Este tipo de inconvenientes se reflejan en proyectos que finalizan sin alcanzar la meta establecida, sin cumplir con los requerimientos del cliente o excedidos en tiempo y presupuesto entre otras varias circunstancias.

Un estudio del Standish Group [1], ejecutado en el 2005, presenta algunos interesantes resultados respecto a proyectos de software, encontrando que solo un 34% finalizan en tiempo y presupuesto, en tanto que un 51% se exceden e inclusive un 15% de ellos es cancelado.

La complejidad del manejo de un proyecto fortalece a estas cifras, lo que la convierte en uno de los mayores retos a superar para un administrador.

En la búsqueda de una receta para disminuir esta complejidad y conseguir proyectos exitosos, es esencial obtener unicidad en el concepto del producto a fabricar, y además procurar difundirlo en todo el equipo de trabajo para conseguir un pensamiento homogéneo respecto a la meta a alcanzar.

Reforzar los métodos, registrar, aprender y retroalimentarse, son también una buena práctica durante la ejecución de un proyecto y también durante la iniciación de nuevos en función de la experiencia adquirida con anterioridad.

Como se puede ver, la comunicación y la organización muestran un camino para dar soporte a quienes lideran los proyectos, por ello en el mercado se han puesto a disposición varias herramientas destinadas a asistir a los proyectos, tal es el caso de Microsoft Project o alternativas Open Source como OPENPROJ, PHPROJEKT, etc., que proporcionan funcionalidades para el registro de actividades, cronogramas, recursos, reportes, relacionadas al quehacer de un proyecto.

Sistemas como estos, han demostrado ser de utilidad, lo que queda entredicho por la cantidad de usuarios con los que cuentan actualmente, sin embargo, la funcionalidad que proveen puede verse enriquecida con la aplicación de nuevas tecnologías que permitan obtener mayor información del entorno y difundirla de formas más adecuadas.

Desde ese punto de vista el presente trabajo de tesis estudia cómo pueden fortalecer, la aplicación de la sensibilidad al contexto, ontologías y folcksonomías, a un entorno de Administración de Proyectos.

La Sensibilidad al Contexto facilitará la obtención de información del entorno en el que se desenvuelve un proyecto, haciendo que el sistema a quien asiste conozca más y pueda sugerir mejores alternativas o adaptarse de forma concordante a la realidad actual en la que se halla. Como es de esperar, mayor cantidad de datos proporcionan información suficiente para ejecutar procesos en tiempos menores y proporcionar funcionalidad más compleja a los sistemas de gestión de proyectos.

En conjunto con la sensibilidad al contexto es dable el uso de ontologías que permitan determinar ciertas reglas sobre el significado de las palabras dentro de la organización. Armar ontologías adecuadas, ayudan a encontrar y clasificar con mayor facilidad y exactitud la información que es suministrada. Se podría armar una ontología para el edificio en la que se especifiquen todos los lugares y determinar si un individuo está dentro del edificio o fuera de él. Las ontologías también serán de utilidad para la búsqueda de material relacionado al proyecto, haciendo nuestro el objetivo de la Web Semántica de conseguir resultados de búsquedas más precisos.

Otro método para alcanzar la satisfacción en los resultados de las búsquedas de material generado con el proyecto, es el empleo de Folcksonomías, a través de ellas es posible que los distintos participantes del proyecto puedan marcar la documentación favorita con etiquetas personalizadas. Esta clasificación, puede difundirse en el equipo de trabajo para facilitar la búsqueda a otros participantes así como al propietario de la etiqueta quien podrá ubicar de forma precisa todos los documentos marcados con cierta etiqueta.

En general, la sensibilidad al contexto, folcksonomías y ontologías son una alternativa válida para conseguir la tan ansiada comunicación y organización que los proyectos solicitan para alcanzar sus metas de forma oportuna y correcta, el presente trabajo pretende generar un modelo que reúna

toda la funcionalidad de estas tecnologías para aplicarlas a un entorno de administración de proyectos.

1.2. Objetivo de la Tesis

1.2.1. Objetivo General

Generar un modelo que permita integrar Sensibilidad al Contexto, Folcksonomías y Ontologías a un entorno de Administración de Proyectos.

1.2.2. Objetivos Específicos

- Determinar cómo la aplicación de Sensibilidad al Contexto, Folcksonomías y Ontologías en un modelo de Administración de Proyectos, pueden ayudar a solventar los problemas reconocidos en los procesos o en su defecto, mejorar los procesos existentes.
- Generar un modelo basado en las oportunidades de aplicación.
- Generar un prototipo que permita verificar la utilidad del modelo diseñado.

Administración de Proyectos

1.3. Introducción

Quien inicia un proyecto, se encuentra con una amalgama de conocimientos y recursos que deberán trabajar en conjunto para alcanzar los objetivos planteados.

En ese caso, encontrar la sinergia, es una tarea compleja que requiere de prácticas capaces de ordenar y dirigir procesos; para solventar esta necesidad, nace la Administración de Proyectos.

Las nociones de la Administración de Proyectos, son referenciadas de manera magistral en el Cuerpo del Conocimiento de la Administración de Proyectos [2], (PMBOK por sus siglas en inglés), editado por el Instituto de Administración de Proyectos (PMI), una organización sin fines de lucro que se dedica a desarrollar la disciplina de Administración de Proyectos en el mundo. Su sede está en Pensilvania, Estados Unidos y tiene más de 219.000 miembros en 125 países.

Al hacer uso del PMBOK, se puede acceder a lo que se ha convertido en los pilares de la Administración de Proyectos, y con ello, evidentemente, lograr más eficiencia en los proyectos actuales, a más de determinar un lenguaje común entre quienes lo emplean.

En las secciones siguientes se describirán, algunos conceptos elementales sobre la Administración de Proyectos así como aspectos requeridos para el entendimiento del presente trabajo de tesis.

1.4. Conceptos

Para comprender lo que es la Administración de Proyectos, vale detenerse en el concepto de proyecto. El PMI, emplea los siguientes términos para referirse:

"Un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único".

Con este concepto, podemos comprender la definición de Administración de Proyectos, según el PMI:

"La Administración de Proyectos es la aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades de un proyecto para satisfacer o exceder los requisitos de los stakeholders de un proyecto".

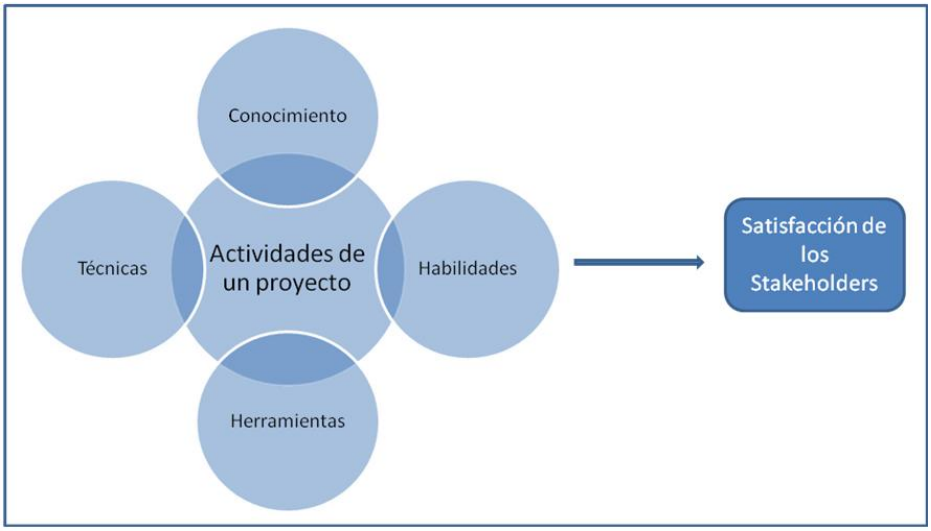


Ilustración 1: Actividades de un Proyecto

Como stakeholders, se conocen a todos los individuos u organizaciones involucradas o eventualmente afectadas por un proyecto.

1.5. Framework PMI

El PMI, establece un Framework para Administración de Proyectos que consta de 9 áreas de conocimiento que al trabajar de una manera integrada y por medio de la correcta aplicación de técnicas y herramientas, consiguen el éxito del proyecto. Información del Framework PMI se puede obtener a través del PMBOK [2].

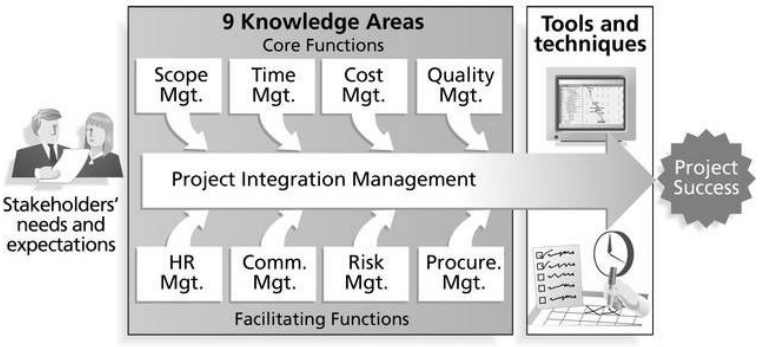


Ilustración 2: Áreas de conocimiento. Fuente: PMI

Las áreas de conocimiento identificadas son:

- Integración
  - Objetivo: Asegurar la coordinación de todos los elementos del proyecto de manera adecuada.
  - Actividades: Entre varías, las principales son:
    - Creación del plan de proyecto
    - Dirección y gestión de la ejecución del proyecto
    - Supervisión y control de cambios
    - Cierre del proyecto
- Alcance:
  - Objetivo: Definir, sin exceder, todo lo necesario por hacer, para completar el proyecto.
  - Actividades:
    - Definición del Alcance
    - Creación de WBS (Work Breakdown Structure)
    - Control y verificación del alcance
    - Control de cambios de alcance
- Tiempo:
  - Objetivo: Asegurar la conclusión del proyecto a tiempo.
  - Actividades:
    - Definición de actividades
    - Estimación de duración y recursos de actividades

        - Desarrollo y control del cronograma
- Costo
    - Objetivo: Asegurar la conclusión del proyecto con el presupuesto aprobado.
    - Actividades:
        - Estimación de costos
        - Preparación del presupuesto
        - Control de costos
        - Control y verificación de los costos del ciclo de vida
- Calidad
    - Objetivo: Asegurar que a la conclusión del proyecto, el mismo satisfaga los requerimientos por los cuales se emprendió.
    - Actividades:
        - Planificación de Calidad
        - Aplicación de actividades de aseguramiento de calidad
        - Control de calidad
- Recursos Humanos
    - Objetivo: Asegurar el uso efectivo del recurso humano involucrado.
    - Actividades:
        - Planificación de los recursos humanos
        - Adquisición del equipo de trabajo
        - Desarrollo del equipo de trabajo
- Comunicaciones
    - Objetivo: Asegurar la generación, reunión, distribución, de la información del proyecto.
    - Actividades:
        - Planificación de las comunicaciones
        - Distribución de Información
        - Información de rendimiento
        - Cierre administrativo
- Riesgos

    - Objetivo: Aumentar la posibilidad de eventos positivos y disminuir la posibilidad de eventos negativos.
    - Actividades:
        - Planificación de la gestión de riesgos
        - Identificación de riesgos
        - Análisis del riesgo (cuantitativo y cualitativo)
        - Planificación de respuestas
        - Seguimiento y control
- Procurement
    - Objetivo: Adquirir, fuera de la organización, los productos, servicios o resultados necesarios para el trabajo del proyecto.
    - Actividades:
        - Planificación de la adquisición
        - Planificación de contratación
        - Selección de proveedores
        - Administración de contratos

Vale la pena hacer hincapié en que el área de conocimiento “Project Integration Management” ayuda a asegurar que todas las demás áreas de conocimiento trabajen de forma coordinada, esto lo consigue al constituirse en la vertebra principal de este Framework a la cual todas las demás áreas se dirigen. Cada una de estas áreas del conocimiento se conforma por procesos, estos procesos se describen a través de inputs, herramientas y técnicas, y outputs.

Existen 44 procesos distribuidos en las distintas áreas y agrupados en:

- Comienzo
- Planeación
- Ejecución
- Control
- Cierre

1.6. Sistemas para la Administración de Proyectos

La complejidad del manejo de un proyecto genera la necesidad de crear sistemas de software capaces de automatizar algunas de sus tareas, razón por la que en el mercado se ha empezado a ofrecer gran variedad de sistemas diseñados para tal propósito.

En este ámbito, se pueden encontrar desde los más básicos hasta aquellos que intentan cubrir todas las áreas del conocimiento de la Administración de Proyectos. Algunos como el MS Project, están bastante difundidos.

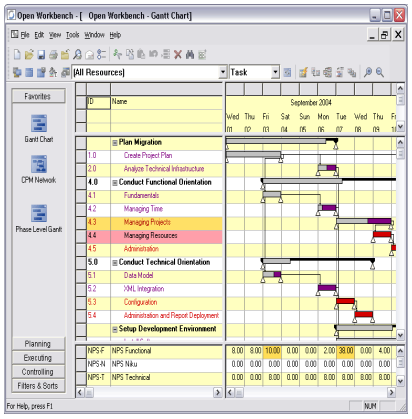
Los sistemas para Administración de Proyectos, pueden ser de código abierto (openSource) o cerrado. Los Open Source permiten acceder a los detalles de su construcción e inclusive realizar mejoras para luego compartirlas a la comunidad. En algunos casos los OpenSource son también de libre uso, es decir no requieren cancelar un valor por su licencia, quedando sin embargo abierta la posibilidad de hacer una donación en beneficio de su desarrollo y mejora.

Estas últimas características hacen que los sistemas OpenSource sean seleccionados por varias instituciones y empresas a la hora de elegir un sistema para administrar sus proyectos.

En las siguientes sub secciones se hace un breve repaso entre los sistemas de software libre más representativos para la Administración de Proyectos.

1.6.1. Open WorkBench

Open WorkBench [3], es un emprendimiento de la empresa CA Clarity, (antes conocida como Niku), que nace como una mejora al sistema Niku Project WorkBench y se posiciona como un sistema de código abierto al cual la comunidad de desarrolladores puede agregarle funcionalidad.



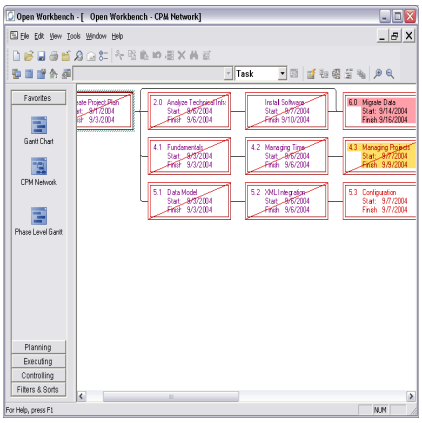


Imagen 1: Algunas pantallas de Open WorkBench

Open WorkBench permite el manejo de proyectos, usando WBS (Work Breakdown Structure) para asociar tareas, actividades, fases e hitos. También contempla el manejo de subproyectos, calendarios, recursos humanos, materiales o de equipamiento, asignación de tareas a recursos, etc.

Entre sus ventajas se puede citar su capacidad para presentar diagramas Gantt así como guiar el cálculo del Earned Value. Sus desventajas se centran en que su única plataforma de funcionamiento es Windows y que requiere de la instalación de algunos sistemas extras (con los cuales tiene interacción), que en su mayoría requieren una licencia paga.

1.6.2. PhProjekt

PHPProjekt [4], está construido en PHP y MySQL como motor de base de datos, ambas herramientas de distribución libre, corre en entornos Web. Su principal intención es permitir la coordinación de actividades, tanto en una intranet como por medio del Internet.

Se distribuye de forma gratuita y su construcción modular alienta a los desarrolladores de PHP a realizar mejoras. Entre sus capacidades están el manejo de calendarios, recursos, reserva de recursos, time card, entre otros. Soporta varios idiomas.

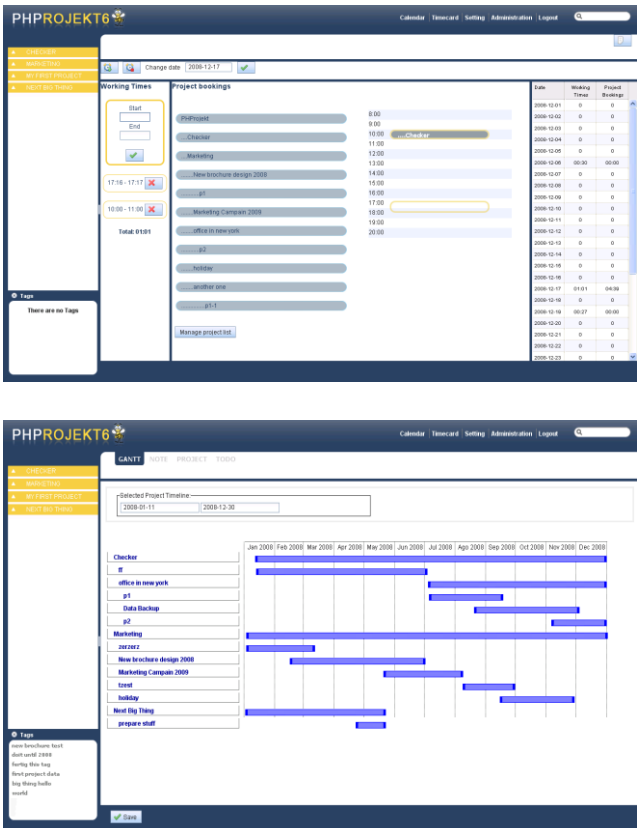


Imagen 2: Algunas pantallas de PHProjekt

Provee además un soporte pago, para acceder a personalizaciones del sistema de tal manera de conseguir un mejor desempeño del mismo. Este soporte es el MyFlower.

1.6.3. OpenProj

OpenProj [5], es un sistema de software diseñado para asistir a la Administración de Proyectos, está construido en Java y se distribuye a través de la licencia: Common Public Attribution License 1.0 (CPAL). Está disponible para sistemas operativos Linux, Windows, OS X.

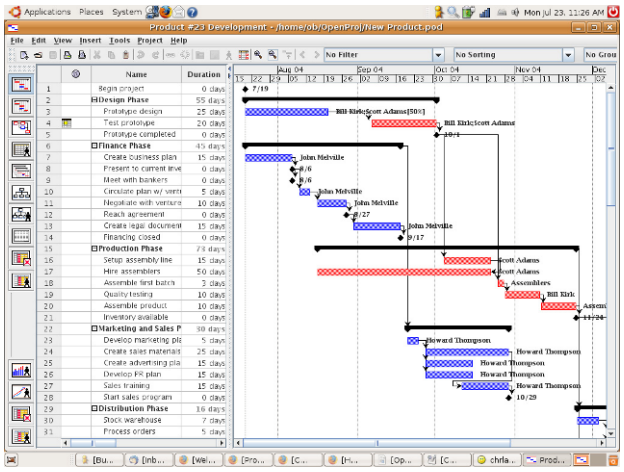


Imagen 3: Pantalla de OpenProj

Projity, una empresa con oficinas en los Estados Unidos, Francia e India, es la responsable de poner a nuestra disposición OpenProj así como el sistema Project On Demand disponible vía Web, que a diferencia del primero, requiere del pago de un pequeño abono mensual para su utilización. OpenProj está disponible en varios idiomas, incluyendo el español.

Entre sus funcionalidades se pueden citar:

- Gestión de Proyectos: Permite manejar información variada sobre el proyecto como nombre, administrador, fechas inicio y fin, tipo de proyecto, etc.
- Gestión de Recursos: Permite el manejo de recursos tanto humanos como materiales, crear grupos de recursos, asignar un calendario de disponibilidad, generar diagrama RBS (Resource Breakdown Structure), establecer costos diferenciados por recurso, reservar personal para tareas etc.
- Gestión de Tareas: Permite manejo de tareas, asignar predecesoras y sucesoras, recursos, WBS, diagramas Gantt, etc.
- Gestión de Calendarios: Permite determinar el número de horas por día, días por mes, etc.
- Gestión de Línea Base para re planificaciones
- Seguimiento del proyecto
- Compatibilidad con archivos de los sistemas MS Project y Primavera, tanto importación como exportación (vía XML)
- Reportes varios

1.7. Algunas oportunidades de mejora de los sistemas para Administración de Proyectos

Las aplicaciones diseñadas para la Administración de Proyectos, brindan una amplia gama de servicios para los líderes, sin embargo se detectan algunas oportunidades de mejora que pueden ser solventadas aplicando elementos de computación sensible al contexto como se describe en el capítulo "Aplicando Sensibilidad al Contexto a Sistemas para la Administración de Proyectos".

Estas oportunidades de mejora, se identificaron después de verificar la funcionalidad de los sistemas para Administración de Proyectos descritos en la sección 2.4 de este capítulo.

Conforme a lo indicado, se pueden anotar las siguientes oportunidades de mejora:

- La creación, modificación, eliminación o finalización de un componente del proyecto debe ser notificada a los interesados por una persona encargada. Si esta persona, olvida informar sobre algún dato de interés (o no lo ha conocido), el proyecto puede tardar en continuar, o continuar en una dirección equivocada, ejemplo de ello son los cambios en la lista de requerimientos, si estos no son notificados a tiempo, podrían generar que el proyecto este avanzando sobre una lista de requerimientos equivocada, provocando retrasos, re-trabajo y elevación de costos.
- La información histórica de proyectos anteriores, es usada pobremente en la toma de decisiones de proyectos actuales. Al administrador del proyecto o a algún miembro del equipo le corresponde indagar en la información de proyectos del pasado para buscar pistas que le ayuden a refinar sus procesos. Si el administrador del proyecto no averigua lo necesario o simplemente no usa información anterior, el nivel de mejora disminuirá para los proyectos.
- No todos los productos del proyecto se integran en la herramienta de Administración de Proyectos de forma ordenada y con trazabilidad. Este inconveniente, a veces provoca el empleo de varias herramientas para la gestión del proyecto.
- Se desarrollan en un entorno ajeno o independiente de su contexto actual, lo que trae consigo que siempre su funcionalidad sea la misma, sin importar las características de la situación actual en la que están siendo empleados.

Aplicaciones Sensibles al Contexto

1.8. Introducción

Analizar información implícita es una capacidad que por mucho tiempo ha estado restringida a la inteligencia humana.

Ante una determinada situación, el cerebro nos suministra de información relacionada al episodio actual, proveniente ya sea del entorno o de experiencias almacenadas en la memoria. Con toda esa información al alcance, somos capaces de, por ejemplo, deducir, actuar de una manera particular, aprender, realizar actividades inconscientemente, entre otras.

Esta capacidad, es la que se pretende dotar a los sistemas de software a través del uso de la Sensibilidad al Contexto (Context - Aware). De esta manera las aplicaciones pueden conocer el contexto en el que se desenvuelven y en base a ello determinar una reacción coherente y fundamentada no solo en lo que explícitamente conocen, sino también en función del análisis de su contexto actual.

Una aplicación que es sensible al contexto, en efecto es una aplicación más inteligente, debido a que captura de forma automática el contexto, lo transforma en datos útiles y finalmente, analiza y toma decisiones sobre su accionar.

En el caso del ser humano, la adquisición de la información es trabajo de sus sentidos y de aquello que permanece en su memoria. Un sistema de software, por su lado, requiere hacer uso de dispositivos, agentes o de otros sistemas de software que se encuentren ubicados en diversos lugares trabajando permanentemente en obtener e informar sobre los cambios contextuales que se den.

En general, la sensibilidad al contexto transfiere a los sistemas de software la oportunidad de deducir comportamiento, al analizar no solo información que está debidamente explicitada, sino también aquella que es implícita.

El resultado de su aplicación es el acceso a más y mejores servicios y por sobre todo: servicios consistentes con su contexto actual.

1.9. Inicios

En 1991 se publicó "The Computer for the 21st Century" de Mark Weiser, donde se idealiza un mundo en el que las computadoras están presentes en todas partes, intentando pasar desapercibidas pero a su vez, sin descuidar el propósito de brindar servicios que beneficien al hombre.

Weiser enfatizó en el hecho de conseguir que los dispositivos de cómputo sean transparentes al usuario, ya que de esa manera, el humano sería beneficiario de servicios apropiados sin tener que interactuar conscientemente con los computadores, esta facilidad redundará en el éxito de lo que Mark Weiser llamó: Computación Ubicua. Las primeras líneas de su artículo dejan claro su objetivo:

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." [6].

La idea de Weiser, dio pie a diversas disciplinas encaminadas a conseguir que las máquinas se adapten al entorno humano convirtiéndose en elementos casi imperceptibles con el propósito de obtener beneficios de ellas constantemente.

Entre estas disciplinas se cuenta a la Sensibilidad al Contexto.

Context – Aware, o su traducción al castellano, Sensibilidad al Contexto, fue nombrada por Schilit [7] en 1994 al referirse a sistemas capaces de adaptarse conforme al contexto en el que se encuentran.

A partir de este punto, se iniciaron distintas investigaciones en pos de obtener soluciones a los desafíos que plantea lo que está llevando a acercarse cada vez más al mundo que Weiser predijo.

1.10. Contexto, Aplicaciones Sensibles al Contexto y Tipos de Información Contextual

El contexto de una aplicación para Dey y Abowd [8], es cualquier información que pueda usarse para caracterizar la situación de una entidad. La entidad es cualquier persona, lugar u objeto que se considera importante para la interacción entre el usuario y una aplicación, incluyendo al usuario y la aplicación.

Nótese, que esta definición considera que no es posible determinar cuáles son los aspectos importantes de todas las situaciones, ya que por cada situación la relevancia puede cambiar.

Supongamos el caso de dos personas que comparten su afinidad por visitar un centro comercial conocido de la ciudad pero que difieren en sus lugares de trabajo.

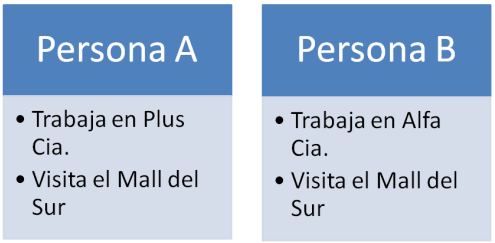


Ilustración 3: Situación de dos personas

La compañía Alfa, posee un sistema de software sensible al contexto que suministra a sus empleados información sobre la agenda para el día de la fecha. El Mall del Sur, también tiene desarrollado un sistema sensible al contexto orientado a proporcionar a sus clientes información sobre ofertas en las tiendas del centro comercial.

¿Qué aspectos son importantes en cada situación?, examinemos el software de la compañía Alfa: éste requiere proporcionar información exclusivamente a sus empleados por lo cual el aspecto de identidad y ubicación es relevante, cuestión que no interesa al software del Mall del Sur, pues este provee información sobre ofertas a cualquier individuo que ingrese a sus instalaciones.

En Mall del Sur		**En Alfa Cia.**	
Información que recibe:		Información que recibe:	
Persona A	**Persona B**	**Persona A**	**Persona B**
Oferta en tienda All's	Oferta en tienda All's	Bienvenida a Alfa Cia.	9:00 Reunión Cliente A
2x1 en Super Cinema	2x1 en Super Cinema		11:00 Capacitación en administración
$10 hamburguesa en Chesse	$10 hamburguesa en Chesse		13:30 Almuerzo con cliente Promerica
6 meses sin intereses en Confort	6 meses sin intereses en Confort		16:00 Visita al cliente Domus

Tabla 1: Ejemplo de aplicación de Sensibilidad al Contexto

Una aplicación es sensible al contexto si emplea el contexto para adaptarse a las necesidades del usuario.

Por otro lado, existen taxonomías o clasificaciones de varios autores que se refieren a la información contextual, de entre las cuales la más aceptado es la de Abowd y Dey [8]:

- Tiempo
- Identidad
- Actividad
- Localización

Cada uno permite manejar una dimensión distinta del contexto. Al fusionar estos tipos base de contexto es posible conseguir información sustanciosa respecto a la situación actual en la que el sistema está siendo empleado y utilizarla para la construcción de diversos servicios personalizados.

1.11. Uso del Contexto en aplicaciones

Supongamos la existencia de un edificio en el que funciona una entidad financiera. El edificio consta de varias habitaciones, cada una de ellas con un propósito específico como por ejemplo oficinas o salas de reuniones. Carlos es un empleado de la entidad financiera que está recibiendo una llamada a su teléfono celular, mismo que cuenta con un sistema sensible al contexto que analiza las características actuales del contexto de Carlos. El sistema a través del análisis de la identidad, conoce que la llamada va dirigida a Carlos y además sabe que Carlos es empleado de la entidad financiera, luego averigua su contexto de ubicación y conoce que Carlos está en estos momentos en el edificio de su trabajo y a mayor detalle conoce que está dentro de una habitación que tiene el propósito de ser sala de reuniones, el sistema procede a analizar el contexto relacionado al tiempo y conoce que ha esta hora Carlos tiene una cita para una reunión de alta prioridad. Con todos esos datos el software puede deducir que Carlos probablemente no desee atender esa llamada y simplemente ordenará al celular que no suene.

El párrafo anterior, cuenta un ejemplo de aplicación de sensibilidad al contexto, pero los escenarios pueden ser muy distintos en función de la realidad contextual.

Pensemos por ejemplo, si Carlos hubiera estado atrasado a su reunión y en el momento de la llamada de celular, aún se encontraba conduciendo en la autopista trasladándose a su lugar de trabajo, en ese caso el sistema deduciría que la llamada puede ser atendida por Carlos.

Muchas aplicaciones sensibles al contexto están siendo desarrolladas en la actualidad para temas de asistencia al turista. Podríamos pensar en ejemplos como el de un dispositivo móvil que ofrezca un listado de sitios de comida de la preferencia del turista, cercanos al lugar en que se encuentre, a la hora en que acostumbra comer, siendo lo más importante que estas sugerencias son dadas sin haberse tomado la molestia de preguntárselo explícitamente a su software.

Los usos de la sensibilidad al contexto, pasan por mucho más que ofrecernos un lugar cercano para comer, en realidad no tienen una frontera establecida, pues muchos de sus servicios son aplicables en casi todas las áreas, lo que se evidencia en algunos ejemplos de alto nivel citados a continuación:

- Hypermedia Adaptativa: Estas aplicaciones mantienen como objetivo la adaptación del sistema al usuario. Brusilovsky [9], en el 2001, se refirió a los sistemas de Hypermedia Adaptativa como aquellos que construyen un modelo a través de la interacción con el usuario, para conseguir adaptarse a dicho usuario. La sensibilidad al contexto, puede entonces, asistir a conocer quién es el usuario, dónde o qué está haciendo, de esta manera provee la información necesaria para que la Hypermedia Adaptativa construya sus modelos resultando en una adaptación transparente para el usuario. Un ejemplo de hipermedia adaptativa son aquellos sistemas orientados a la adaptabilidad de contenidos educativos, un caso es el Sistema Hipermedia Adaptativo para contenidos educativos basados en tecnología de agentes de software, que permite adaptar información en función de los estilos de aprendizaje partiendo de un modelo de programación neurolingüística.
- Location-Based Services (LBS) [10]: Se basan en la localización del usuario para proveer servicios. Existen cientos de aplicaciones similares, entre las que se cuentan: Guías Turísticas, Buscadores de Amigos, Información de Transporte Público, Ubicación de lugares cercanos, Servicio Metereológico, etc.
- Hogares inteligentes [11]: El principio de estos hogares es brindar comodidad y servicios a quienes lo habiten. Utilizan información suministrada por una red de dispositivos,

ubicados en el hogar, los mismos que permiten capturar la información del contexto y con ello proporcionar servicios variados relacionados a las actividades comunes en un hogar.

1.12. Arquitecturas

Desde un punto de vista más técnico, los sistemas sensibles al contexto se hallan en vías de desarrollo y han sido sujeto de estudio por varios años. Entre los más representativos estudios están el Context Toolkit [12], Hydrogen [13], etc.

Estos marcos de trabajo definen métodos para obtener los datos que los sensores proporcionan e interpretarlos, además de manejar los servicios a los que se pueden acceder y los mecanismos de suscripción a los mismos.

El resultado ha sido un alto grado de conocimiento sobre la materia y su aplicación con éxito en ciertas áreas, el saldo negativo es que no se ha logrado determinar un estándar a seguir para la construcción de una aplicación Context Aware o UbiComp que provea soluciones a todos los inconvenientes que representa.

Más que limitación tecnológica, el hecho de no estandarizar su construcción se debe a que, la gama de aplicaciones a las que se puede dotar de sensibilidad al contexto, es muy grande y cada una trae consigo una serie de particularidades que hacen complejo definir un único framework de construcción.

Para solucionar este inconveniente el Laboratorio de Investigación y Formación en Informática Avanzada LIFIA de la Facultad de Informática de la Universidad Nacional de La Plata, está estudiando, en lugar de un framework, la creación de una plataforma para desarrollar UbiComp, que sea capaz de incorporar su funcionalidad a aplicaciones que carecen de sensibilidad al contexto.

Esta investigación indica la necesidad de contar con bloques simples de construcción que sean independientes del dominio y que signifiquen una abstracción de los conceptos más comunes que la computación ubicua pregona. Por otro lado estaría el construir frameworks y aplicaciones por encima de estas abstracciones.

La información contextual se distribuye como una mejora a los objetos que residen en el modelo de la aplicación (o se constituyen en los objetos del modelo de la aplicación), dado que consideran que el contexto es un conjunto de pequeñas piezas contextuales.

APLICACIÓN SIN SENSIBILIDAD AL CONTEXTO	APLICACIÓN CON SENSIBILIDAD AL CONTEXTO
Object Persona	**Aware Object** Persona
- Conoce su nombre - Conoce su apellido - Conoce su dirección	- Conoce su nombre - Conoce su apellido - Conoce su dirección - Conoce qué hacer a determinada hora del día

Tabla 2: Aware Objects

La tabla 2, presenta dos objetos Persona, el primero (aquel que pertenece a una aplicación sin sensibilidad al contexto), conoce únicamente sus características conceptuales, como su nombre o su dirección, dicho de otra manera, es el típico objeto estudiado en el diseño orientado a objetos. A este tipo de objetos se los conoce como Object en Context-Aware. En el segundo caso se presenta al mismo objeto creado en una aplicación sensible al contexto, si se observa, su mayor diferencia, es que adicional a sus características conceptuales, conoce como reaccionar a determinada hora del día, o dicho de otra forma, tiene capacidad de razonar en función a su contexto pues conjuntamente a sus atributos conceptuales maneja atributos contextuales. Esta clase de objetos se los conoce como Aware - Object: objetos capaces de manipular su contexto.

Los Aware – Object, se emplean en aplicaciones que desde sus inicios son diseñadas para trabajar en entornos sensibles al contexto, sin embargo, la realidad actual significa un amplio conjunto de aplicaciones que no fueron diseñadas para interactuar con su contexto, intentar realizar una reingeniería de todas ellas podría llegar a ser costoso, lento e inclusive inaplicable, es por ello que se propone el uso de un tipo de objetos denominado Aware – Model: objetos que permiten extender Objects de tal manera que puedan manejar su contexto y mantener su funcionalidad actual.

APLICACIÓN SIN SENSIBILIDAD AL CONTEXTO	APLICACIÓN ADAPTADA PARA SENSIBILIDAD AL CONTEXTO
Object Persona	**Aware - Model**

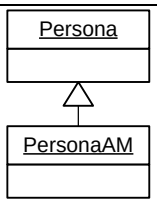
	
- Conoce su nombre - Conoce su apellido - Conoce su dirección	- El Objeto PersonaAM extiende a Persona - Persona conoce su nombre, apellido, dirección - PersonaAM conoce qué hacer a determinada hora del día

Tabla 3: Aware Models

Incorporar en las aplicaciones los Aware – Model, es una manera ingeniosa de mantener la funcionalidad actual de los objetos del software y crear nuevas instancias competentes para manipular su contexto. Esta solución permite agregar sensibilidad al contexto sin necesidad de desechar clases o reconstruir el software por completo.

Dentro de este ámbito podemos hablar de los siguientes significados:

- Aware Object: Es un objeto capaz de manipular su contexto.
- Context Feature: Es un aspecto específico del contexto de un objeto. Estos se construyen a través de una jerarquía en la que los Context Feature son abstractos y se definen los Model Based Fatures, Tracked Features y Derived Features. Con esto se puede obtener objetos Context Feature más ricos semánticamente y por tanto con más funcionalidad.
- Aware Model: Son objetos que permiten extender objetos de aplicación de tal manera que puedan manejar su contexto y mantener su funcionalidad actual.
- Enviroment: Son ambientes de adaptación encargados de proveer una forma específica de comportamiento una vez que se conoce el contexto actual.
- Handlers: Son objetos capaces de suscribir Aware Objetct o Aware Model a un evento y reaccionar a través de un comportamiento específico.

Los enviroments perciben cambios en el contexto y delegan a otros objetos el hecho de adaptarse. En general los handlers son a quienes se delega esa tarea.

El camino para llegar a una verdadera computación omnipresente en la vida del ser humano, es todavía un escenario que requiere de una decidida investigación multidisciplinaria. La meta

constante, sin duda, es ofrecer facilidades tecnológicas que simplifiquen la vida del ser humano, respetando siempre su individualidad y capacidad de razonamiento.

Ontologías

1.13. Concepto

Según la Real Academia de la Lengua Española [14], ontología es:

"La parte de la metafísica que trata del ser en general y de sus propiedades trascendentales."

En informática, según Studer [15], una ontología es:

"Una especificación explícita y formal de una conceptualización compartida."

Éste árbol de conceptos puede ser leído y procesado por una computadora. De esta manera, los programas informáticos pueden utilizar las ontologías para una variedad de propósitos relacionados con la resolución de problemas por medio del razonamiento inductivo.

Es por ello que su principal aplicación en nuestro campo ha estado atada al mundo de la inteligencia artificial.

1.14. ¿Por qué utilizar ontologías en el software?

Las ontologías al ser independientes del lenguaje y entendibles por las computadoras, son útiles porque ayudan a alcanzar un entendimiento común de la información descriptiva. Este entendimiento puede lograrse entre personas, entre agentes de software, y entre personas y agentes de software.

Además nos permiten reutilizar el conocimiento adquirido del dominio, hacen más fácil modificar las suposiciones sobre un dominio, y nos permiten separar el conocimiento del dominio del conocimiento operacional.

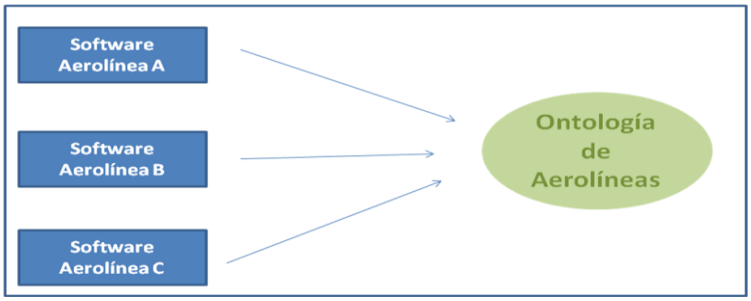


Ilustración 4: Reutilización de una ontología por varios sistemas

En la ilustración, se puede apreciar cómo el software de diferentes Aerolíneas, comparten el uso de una misma ontología, este acuerdo permitirá, en caso de ser necesaria una integración de los sistemas, que todos conozcan y entiendan los diversos elementos de la misma forma. Por ejemplo, para todos, el término pasajero, significará una persona que ha comprado un boleto de un vuelo.

Las ontologías deberán estar bien diseñadas y bien definidas, para ello es indispensable el concurso de expertos en el dominio que define la ontología. Una ontología bien diseñada y definida quiere decir que deberá capturar adecuadamente el dominio modelado, y ser entendible tanto por humanos como por máquinas al lograr proveer un buen soporte para su procesamiento. Además que deben estar bien definidas no solo en su sintaxis sino también en su semántica (significado).

1.15. Elementos de una ontología

Las ontologías para su definición, hacen uso de los siguientes elementos [16]:

- Clases: ideas a formalizar, representan los conceptos en el sentido más amplio.
- Relaciones: permiten la representación de las interacciones entre clases.
- Funciones: casos especiales de relaciones, donde para identificar elementos se requiere del cálculo de una función.
- Instancias: permiten la representación de un individuo.
- Axiomas: permiten representar conocimiento que no puede ser formalmente definido a través del uso de los anteriores elementos.

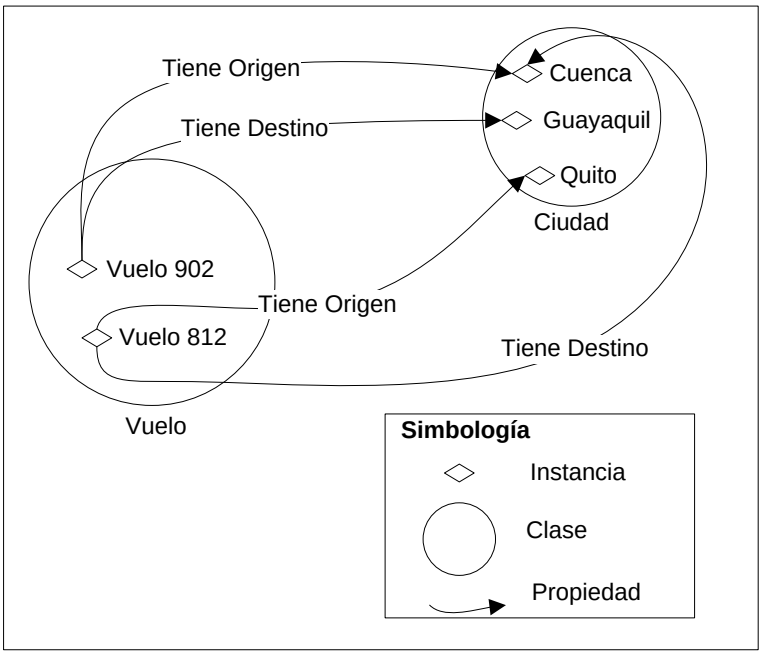


Ilustración 5: Elementos de una Ontología

La ilustración 5 permite apreciar una representación gráfica de una ontología, en la que participan clases, propiedades e individuos. La lectura de la ilustración nos indica, por ejemplo, que el vuelo 812 parte de la ciudad Quito con rumbo a la ciudad Cuenca, de esta forma, la clase vuelo y ciudad, tiene además de un significado propio, una relación entre ellas que agrega semántica al dominio modelado.

1.16. Lenguajes ontológicos

Las ontologías, como se indicó en los apartados de este capítulo, son conceptos que pueden ser entendidos y compartidos entre seres humanos, agentes de software o una mezcla de los dos tipos, en ese sentido es necesario establecer un lenguaje para describir las ontologías de la manera menos ambigua posible.

Existen algunos lenguajes orientados a especificar una ontología de un dominio, entre los que cuentan:

- RDFS
- OWL
- DAM - OIL

1.16.1. Resource Description Framework RDF

RDF [17], significa "Resource Description Framework" o marco de trabajo para describir recursos. Es una recomendación candidata W3C y es una forma gráfica basada en sintaxis XML para representar metadata, que permite describir las semánticas de la información en una forma entendible también por computadoras.

A las sentencias se las define como tripletas que tienen la forma <sujeto, predicado, objeto>. Por ejemplo la tripleta: <Vuelo, tieneOrigen, Ciudad>, puede ser representado por el gráfico:

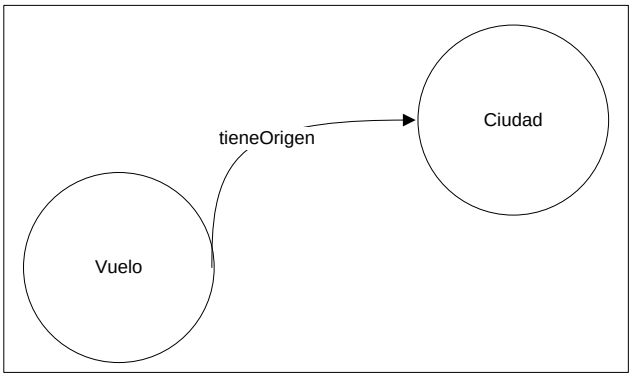


Ilustración 6: Representación de una Tripleta

Las tripletas describen propiedades de recursos. Un recurso es cualquier objeto que puede ser apuntado por medio de una URI (un documento, foto, párrafo), las propiedades por sí mismas son también recursos. URI significa: Identificador de Recurso Uniforme. Es un conjunto genérico de todos los nombres y direcciones que son cadenas de caracteres cortas y que se refieren a recursos.

URL significa: Localizadores de Recurso Uniforme. Son un tipo particular de URI, usado para recursos que pueden ser accedidos en la WWW (Ej. Sitios Web).

RDF tiene una sintaxis XML que posee un significado específico:

- Cada elemento de descripción describe un recurso.
- Cada atributo o elemento anidado dentro de una descripción es una propiedad de ese recurso.
- Podemos referirnos a los recursos usando URIs.

```
<Description about = "miProyecto/vuelo">
        <tieneOrigen resource = "miProyecto/ciudad">
</Description>
```

Ilustración 7: Uso de URIs

RDFS extiende al RDF con vocabulario semántico, como:

- Class, property
- Type, subClassOf, subPropertyOf
- Range, domain

Entre las desventajas de RDF se cuenta su debilidad para describir recursos con el suficiente detalle. No se permiten restricciones "localizables" de rango y dominio.

Por ejemplo, no se puede decir que el rango "Tiene Hijos" es de la clase ontológica Persona cuando se aplica a personas y Elefante cuando se aplica a elefantes.

No se permiten restricciones de existencia, cardinalidad. Por ejemplo, no se puede decir que todas las instancias de persona tienen una madre que es también una persona, o que las personas tienen exactamente dos padres.

No permite transitividad, inversión o propiedades simétricas. Por ejemplo, no se puede decir que "Es parte de" es una propiedad transitiva, que "Tiene parte" es la inversa de "Es parte de" o que "Tiene contacto con" es simétrica.

1.17. Ontology Web Language OWL

OWL [18], Ontology Web Language, lenguaje de ontologías para la web, es considerado como una extensión de RDF. La mejora se puede definir como la capacidad de expresar axiomas formales y propiedades más complejas entres clases ontológicas. Es también manejado por la W3C, con la intención de conseguir la estandarización en el manejo de las ontologías.

OWL, se divide en tres sublenguajes: OWL Lite, OWL DL y OWL Full, este último es el que contiene toda la capacidad expresiva de OWL, sin embargo, puede llegar a ser demasiado complejo para aplicar un razonador (en efecto, requiere mucho trabajo de procesador), por ello se especificaron

versiones que son un subconjunto de las características de OWL Full. OWL Lite es el más ligero, contiene todas las restricciones de OWL DL y algunas más. OWL DL, con menos restricciones que OWL Lite, tiene la intención de facilitar la aplicación de un razonador de lógica descriptiva de manera eficiente.

Según quienes trabajan con ontologías, una buena manera de iniciar es justamente con la aplicación de OWL Lite, para hacer pruebas básicas de la ontología y aplicar pequeños niveles de razonamiento, luego, y mientras se va mejorando la ontología, es recomendable avanzar hacia OWL DL e inclusive OWL Full.

1.18. Algunas Aplicaciones

Además de haber sido utilizadas en la inteligencia artificial, las ontologías son el corazón mismo de la Web Semántica.

La Web Semántica es un proyecto por el cual se busca catalogar la información de los recursos Web, dejando de usar como hasta el momento, palabras clave (keywords) y empezando a utilizar ontologías, es decir, que estas palabras tengan significado.

Con las ontologías, la información respecto al contenido de una página (la metadata) se organizará de manera que los agentes de software (programas que interpretan esta información), podrán buscar e integrar los datos mucho mejor que ahora.

Gracias al conocimiento almacenado en las ontologías, las aplicaciones podrán extraer automáticamente datos de las páginas Web, procesarlos y sacar conclusiones de ellos, así como tomar decisiones y negociar con otros agentes o personas.

Como ejemplo, podemos pensar en un estudiante de historia a quién se le ha indicado la realización de un trabajo sobre la historia del Ecuador. El estudiante, accede al Internet y busca sobre Valdivia (una de las culturas preincaicas de ese país), un buscador web actual basado en palabras claves, presentará resultados muy variados, como por ejemplo enlaces a páginas que hablen de Valdivia: la ciudad ubicada a orillas del Pacífico o de Valdivia: un restaurante de comida típica del litoral, cuestiones que para el caso, no tiene relevancia.

En efecto, a pesar de que los buscadores de la web actual son poderosas herramientas para la localización de información en Internet, sus resultados no son del todo precisos quedando la tarea

de escudriñar entre los enlaces devueltos al usuario hasta ubicar un enlace que satisfaga su interés. Si el estudiante del ejemplo, empleara para su investigación un buscador semántico podría ubicar su búsqueda exclusivamente al dominio de la historia, y más específicamente a la historia del Ecuador, en donde la palabra Valdivia tiene el significado de cultura preincaica y obtener enlaces a páginas con información únicamente de Valdivia en ese contexto. Por otra parte un buscador semántico analizaría el significado de la palabra Valdivia y podría brindar al usuario alternativas de navegación como páginas relacionadas a otras culturas preincaicas del Ecuador, también podría conocer que Valdivia fue una cultura muy desarrollada en la producción de productos cerámicos y podría ofrecer alternativas de enlaces a páginas con imágenes sobre restos arqueológicos de esta cultura, etc.

Una ontología especifica formalmente un dominio determinado, por lo que sus aplicaciones pueden ser muy variadas. Una ciudad puede ser descrita a través de una ontología, en la que la clase ontológica “ciudad” se relaciona con la clase “lugares”, a su vez la clase “lugares”, define varias subclases como “lugar de interés turístico”, “lugar de interés local”, “lugar de interés político”, etc., cada subclase tendrá un significado distinto. Esta ontología podría emplearse para definir el comportamiento de un software sensible al contexto que se adapte al usuario según su ubicación. Si un turista está visitando una ciudad determinada y su ubicación actual es justo en uno de los lugares de interés local (como un sitio de recaudación de impuestos), la ontología podría analizar el concepto “lugar de interés local” y el concepto “turista” para concluir que probablemente al turista no le interese conocer ningún detalle sobre ese sitio. La situación sería contraria si el turista estuviera en un “lugar de interés turístico” cuya propiedad “interesados” cuenta a los turistas.

La iniciativa de la generación de ontologías, pretende formalizar el conocimiento del ser humano en diversos ámbitos y bajo distintos contextos. Su desarrollo puede ser un trabajo completamente aislado de las tecnologías de la información, aunque su informatización por un lado, permite la generación de un sinnúmero de beneficios a los usuarios de los sistemas, y por otro facilita la construcción y validación de modelos ontológicos. El gran poder de las ontologías, y de paso de las aplicaciones que las emplean es el compartir el conocimiento a través de la aceptación de un concepto por todos los interesados en un dominio.

Folksonomías

1.19. Introducción

La WEB contiene gran cantidad de información sobre distintas áreas del conocimiento humano. Encontrar los datos que se requieren, podría ser una tarea casi imposible si no fuera por la existencia de mecanismos tales como los motores de búsquedas, que han desarrollado algoritmos de alta complejidad para presentar las páginas que más se aproximen al requerimiento de quien está realizando la búsqueda.

Los resultados que podemos obtener al usar un motor de búsqueda, no son completamente precisos, razón por la que es necesario usar el discernimiento humano para hallar entre los resultados, aquellos que realmente satisfagan sus necesidades.

Esta falta de precisión en parte se debe a que en los inicios de la WEB se descuidaron detalles que permitieran ordenar la información de alguna manera. Con el afán de intentar organizarla y de que los resultados de nuestras búsquedas sean más exactos, aparecen dos grandes corrientes: Web Semántica y Web Social.

La Web Semántica es una investigación que realiza Tim Bernerns Lee, que es muy relacionada al concepto de ontologías (vea el capítulo sobre Ontologías para más detalles), las mismas que permiten clasificar no solo por palabras claves, sino por el significado de las palabras que forman parte del contenido, lo que genera un mayor nivel de exactitud.

Por su parte la Web Social, usa un método de tagging o etiquetado, que es aplicado por los usuarios de la red. Cada persona puede etiquetar una página de acuerdo a su criterio o percepción respecto al contenido de la misma. Las tags que recibe un contenido se comparten a la comunidad, lo que ayuda a su clasificación y como efecto de ello están los resultados de búsquedas más cercanos a lo que realmente se está buscando. La desventaja de este método de clasificación es la existencia de una diversidad de tags para catalogar a un mismo contenido.

1.20. Concepto

En el sentido etimológico, folksonomía es un término relacionado a Volks (pueblo en alemán) y Taxonomía (clasificar). Su utilización por primera vez fue hecha por Thomas Vander Wal [19]. Folksonomía pertenece a la Web Social, y se refiere a la clasificación gestionada por los usuarios, o más precisamente por el pueblo.

El proceso de esta clasificación se resume así:

Un usuario etiqueta una página con algún término que la relacione directamente, en una instancia posterior, cuando el usuario desee acceder nuevamente a esa página, podrá buscarla directamente por el tag que colocó previamente. Además puede compartir su tag con la comunidad, logrando que otras personas puedan ubicar el contenido de acuerdo a la etiqueta que el consignó.

Una particularidad radica en la organización no jerárquica de las etiquetas, es decir todas están a un mismo nivel; punto importante a la hora de encontrar diferencias con las ontologías.

1.21. Folksonomías en Internet

La red Internet, es el lugar que mayor aplicación para folksonomías ofrece. Su multitudinaria audiencia, permite conseguir gran cantidad de material clasificado.

Joshua Schachter, es el creador del sitio web pionero en esta rama al que ha denominado del.icio.us [20]. Este gran bookmark, permite a los usuarios registrarse a sus servicios y con ello obtener herramientas que le facilitan el trabajo de etiquetar páginas mientras navegan por la red, luego a través de esas marcas, los usuarios podrán encontrar más rápidamente la página de su interés. Su mayor poder se encuentra en la posibilidad de compartir las clasificaciones con otros usuarios que podrían o no estar registrados a los servicios de Del.icio.us., esto hace que quienes lo conocen en muchas ocasiones lo utilicen como una alternativa a motores de búsqueda como Yahoo o Google.

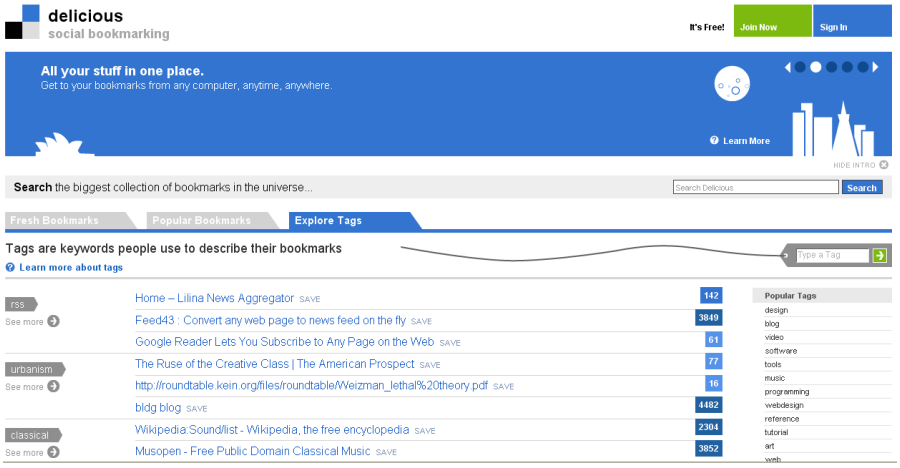


Imagen 4: Página Del.icio.us

En cierta medida, Del.icio.us, mejora la precisión de los resultados de la búsqueda, dado que las etiquetas que proveen los usuarios son producto de su percepción respecto al tema, dicha percepción en muchas ocasiones podría ser la misma de otro usuario, lo que significa que los resultados sean más acotados y precisos. Además ofrece listados que presentan las tags más populares de la red, y de tags relacionadas a la actual.

Otra ventaja de este sitio, es su poder de movilizar las características funcionales de "favoritos" de un browser, al permitir acceder al listado de sitios de interés (aquellos a los que les consignó una tag), en cualquier computador del mundo que esté conectado a la red Internet.

Las folcksonomías, además se aplican a sitios que se encargan de difundir material fotográfico o de video, como es el caso de Flickr [21].

Flickr, actualmente pertenece a Yahoo, y permite a sus usuarios hacer tagging de sus fotografías y en similitud a Del.icio.us, también ofrece un servicio para realizar búsquedas basadas en etiquetas.

Este sistema, como la mayoría de su estilo, es objeto de agrado de cientos de navegantes dando paso a grandes sociedades virtuales, en las que compartir fotografías, páginas de interés o videos, enlaza a personas de todo el mundo.

Esta creciente popularidad, lleva a grandes de Internet como Google, a plantearse la necesidad de construir sistemas que empleen folcksonomías, de tal manera de poder competir con sistemas como Del.icio.us o Flickr.

1.22. Ventajas y Desventajas

A continuación una lista de ventajas que las folcksonomías poseen:

- Método de clasificación gestionado por el pueblo:
    - No requiere personal técnico para esta tarea.
    - Es posible clasificar material antiguo sin necesidad de modificar su estructura.
    - Facilita el etiquetado que más se acomode a la percepción del usuario sobre el tema.
    - Un mismo elemento puede tener varias clasificaciones, consignadas por distintos usuarios.
    - Las clasificaciones pueden obedecer a características particulares de quien etiqueta, o a características de su entorno, por ejemplo el idioma nacional.
    - La clasificación de un elemento por distintos individuos, ayuda a generar tags sinónimas, lo que implica que se podrá obtener una red de relación entre tags.
- Etiquetas compartidas:
    - Cualquier usuario puede acceder a un elemento a través de la etiqueta que le consignó algún usuario en un tiempo anterior.
    - Elementos etiquetados previamente por alguien, podrían ya no necesitar un tag, si uno de los que posee está de acuerdo a la percepción del usuario actual.
    - Creación de sociedades virtuales.
- Resultados de Búsquedas más acotados:
    - La búsqueda a través de tags, puede llegar a ser muy precisa.
    - Los procesos de búsqueda se resumen en buscar la tag buscada.
    - El tiempo requerido para hallar lo que se busca, se puede ver reducido en función de que su clasificación previa le otorgo una etiqueta correctamente descriptiva.

A pesar de tener una buena cantidad de ventajas, las folcksonomías también tienen cierto tipo de limitaciones o problemas:

- Clasificación no guiada:
    - Genera una clasificación desordenada
    - Genera una clasificación muy específica o general

- Clasificaciones que son útiles solo para el usuario que las otorgó
- Etiquetas con faltas ortográficas
- Etiquetas no acordes al contenido

- Acceso al servicio:
  - Los usuarios requieren suscribirse a un servicio de tag
  - Las herramientas de tagging que proveen los sitios con soporte de folcksonomías, por lo general deben ser instaladas en el navegador, por lo tanto estará disponible solo en el computador de instalación con lo que el usuario no podrá taggear páginas desde un computador distinto sin antes realizar el proceso de instalación.

Aplicando Sensibilidad al Contexto a sistemas para la Administración de Proyectos

1.23. Introducción

En la sección 2.5 “Algunas oportunidades de mejora de los sistemas para Administración de Proyectos” del capítulo “Administración de Proyectos”, se pudo detectar algunas limitaciones que poseen los sistemas de software creados para asistir a la Administración de Proyectos y se sugirió que algunas podrían ser resueltas mediante el empleo de computación Sensible al Contexto.

Con la sensibilidad al contexto, los sistemas de Administración de Proyectos estarán capacitados para conocer su entorno y actuar coherentemente con él. Esta característica también va a facilitar la captación de información relacionada al proyecto desde los diversos entes interesados. De esta manera el software no va a depender en su totalidad de la intervención humana para mantener información actualizada sobre el avance del proyecto o las novedades que ocurran.

El sistema podrá conocer la identidad de quien lo opera y proveer servicios personalizados. La dotación de Sensibilidad de Contexto, proporciona una mirada global del entorno en el cual se desenvuelve el proyecto lo que facilitará el manejo de un sistema de alertas para sus usuarios.

Además, si los aplicativos para Administración de Proyectos se desenvuelven en un entorno de computación ubicua será más sencillo que sean empleados o consultados por todos los stakeholders del proyecto.

En el presente capítulo se analiza cómo la Sensibilidad al Contexto puede usarse para beneficio de los sistemas de Administración de Proyectos.

El resultado del análisis es un modelo de clases que soporta la solución a varias necesidades de los sistemas para Administración de Proyectos y un prototipo de software cuyos detalles se los pueden ver en el capítulo “El Prototipo”.

1.24. Integrando Sensibilidad al Contexto

La Administración de Proyectos es una actividad que está estrechamente relacionada con los requerimientos o necesidades que generan un proyecto, con las actividades planificadas y finalmente con el resultado de esas actividades: el producto.

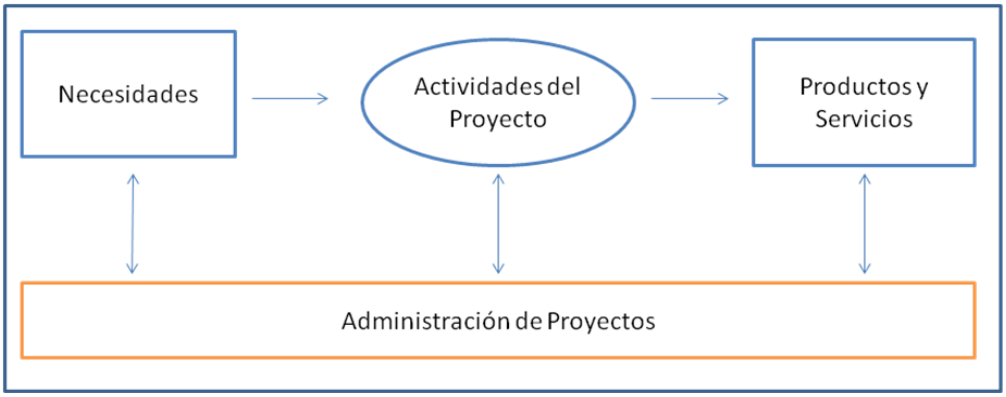


Ilustración 8: Relaciones de la Administración de Proyectos

En medio de esta interacción, existen distintos contextos que de una u otra forma llegan a ser de importancia para el proyecto, ya que ante su ocurrencia se podría actuar de una manera adecuada para obtener mejores resultados en la gestión de un proyecto.

Emplear el contexto de las Necesidades, por ejemplo, ayudará a mejorar la capacidad de los sistemas de Administración de Proyectos para automatizar, de alguna manera, la atención que se requiere sobre los cambios en los requerimientos, liberando al administrador del proyecto de esta actividad.

Sin embargo, hay que considerar, que las Necesidades son abstractas y existen tan solo en el entendimiento, lo que dificulta decir que posean un contexto al cual acudir para obtener información. Este conflicto se supera al definir que cada necesidad posee una fuente y se asume como el contexto de la Necesidad, al contexto de la fuente que la genera.

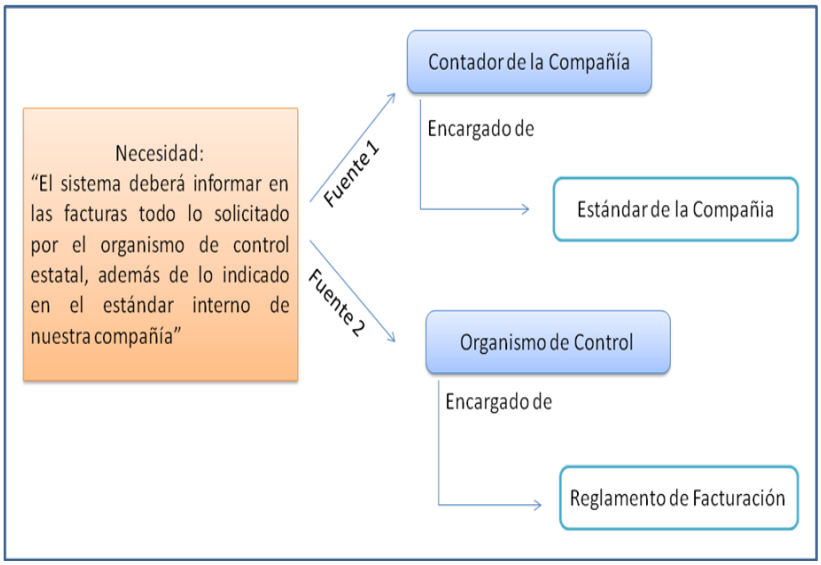


Ilustración 9: Fuentes de las Necesidades de un Proyecto

En el ejemplo de la ilustración, se puede ver como una determinada Necesidad de un proyecto se basa en dos documentos de interés: Estándar de la Compañía y Reglamento de Facturación. El primer documento está a cargo del contador y el segundo es un documento oficial de un organismo estatal de control. El contexto de estos documentos será el contexto de actividad de sus fuentes, de tal manera que cuando el contador realice un cambio al Estándar de la Compañía, se captará un cambio en su contexto de actividad y se podrá informar al respecto a quien corresponda. Es evidente que un analista funcional del proyecto, va a estar muy interesado en conocer los cambios que el contador pueda efectuar en el Estándar de la Compañía, razón por la que se puede definir al analista funcional como destinatario de este tipo de notificaciones.

El contexto de las Actividades, será de utilidad en especial para aquellas que requieren una supervisión continua o que necesitan información que se obtuvo en otro proyecto u en otro momento.

En general, aplicar la sensibilidad al contexto a las Actividades del proyecto puede traer beneficios que se reflejarán en tiempo, costos y un mejor manejo del proyecto.

Finalmente, cuando se concluye un proyecto se contará con un Producto o Servicio listo para ser empleado en las actividades para las que fue construido. El contexto de dicho producto podría pensarse que deja de ser de utilidad para la Administración de Proyectos ya que es el último eslabón en el proceso y el punto que marca el final de un proyecto. Esa apreciación es del todo

equivocada dado que el éxito de un proyecto también es el éxito del producto generado. Entonces sería valioso obtener información del contexto en el que se desenvuelve el producto y en función de ello, por ejemplo, preparar mejoras sobre su funcionalidad, proponer nuevos servicios o productos, encontrar fallas, entre otros aspectos. Una vez más, estaríamos ante la necesidad de identificar qué cosas del contexto del producto son beneficiosas para nuestros proyectos y enfocarnos en ellas.

1.25. Definiendo los contextos de interés

Cada componente de un proyecto posee contextos que pueden ser de interés para las tareas de administración. Un componente se considera cualquier elemento relacionado con un proyecto, como pueden ser subEtapas, actividades, documentos, etc., en realidad la categorización de los elementos de un proyecto como componentes, dependerá de la visión del líder y de cómo lo configure en el sistema.

El modelo que se está diseñando, debe tener la capacidad de permitir la selección de los contextos de interés para el proyecto por parte de su administrador.

Consideremos la existencia de una clase Proyecto. Un proyecto deberá conocer todas las necesidades, actividades y productos que lo componen.

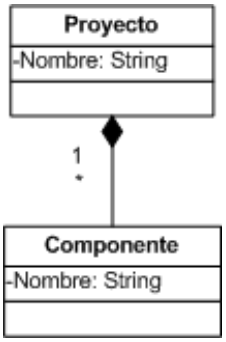


Diagrama 1: Clases Proyecto y Componente

La clase Proyecto conoce su nombre y una colección de componentes de los cuales está formado. La clase Componente podría especializarse en clases para la definición de componentes del tipo necesidad, actividad o producto (como ejemplos), convirtiendo a la clase Componente en una superclase, cuyas subclases heredarían sus atributos y comportamientos.

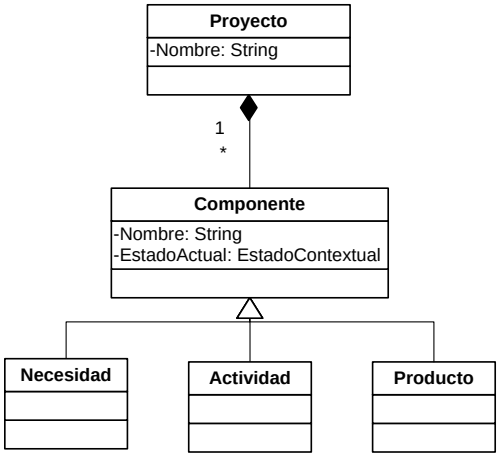


Diagrama 2: Especialización de la clase Componente

Vale la pena detenernos a pensar que no todos los sistemas de Administración de Proyectos integran como un componente del proyecto al producto. En un software factory, los sistemas creados para distintos clientes, en su mayoría, son entregados para su utilización marcándose en ese momento la finalización del vínculo entre sus creadores y su producto. Esta situación genera una brecha entre el proceso que generó un producto y su producto. Con este distanciamiento es dificultoso conocer el desempeño del producto en su operación haciendo que el feedback para proyectos futuros se vea disminuido. De ahí que este modelo sugiere la integración en los sistemas de Administración de Proyectos, de los productos, para lo cual la especialización de la clase Componente en una clase llamada Producto podría ser de utilidad.

Con el esquema de clases planteado en el diagrama 1, es posible crear proyectos y componentes y asignar estos últimos a un proyecto. El administrador del proyecto es el encargado de cargar esta información en los sistemas siendo una tarea cotidiana.

Ahora, para dotar a este esquema de Sensibilidad al Contexto, se va a permitir al administrador de proyectos, seleccionar algunos contextos de interés para el proyecto y definir las acciones a ejecutar cuando ese contexto ocurra.

Hay que considerar que un contexto puede estar relacionado a la identidad, tiempo, actividad o localización de un elemento. En la Administración de Proyectos estos elementos son los componentes y dependerá de su naturaleza los contextos que sean de su interés. En ese sentido,

aparece el término “Estado Contextual”, que es un estado en el que un componente se encuentra cuando un determinado contexto ocurre.

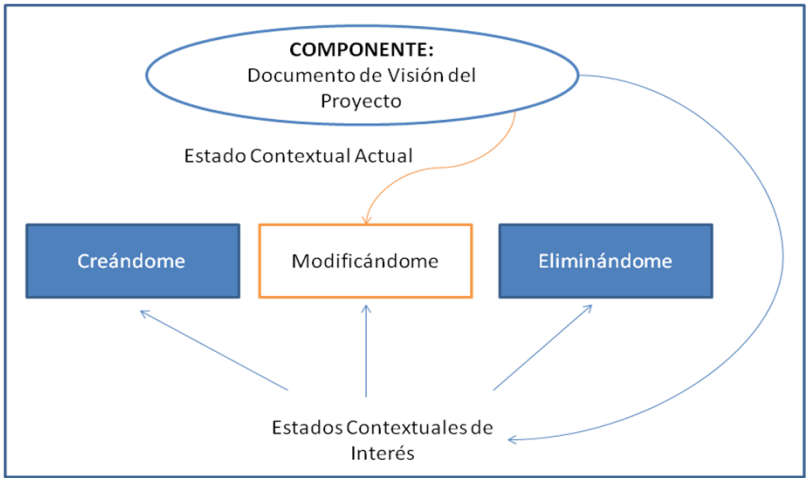


Ilustración 10: Estado Contextual

Para ejemplificar pensemos en el componente “Documento de Visión del Proyecto”, el mismo que puede alcanzar tres estados contextuales de interés para el proyecto (todos relacionados al contexto actividad):

- Creándome
- Modificándome
- Eliminándome

Sin embargo un componente podría estar en un único estado contextual a la vez, por ello se indica que el estado contextual actual es Modificándome, de esa manera el software podrá determinar las acciones a tomar ante la realidad contextual actual. Cuando el estado contextual actual del componente cambie, por decir a, Eliminándome, las acciones a tomar serán aquellas que se enfoquen de mejor manera a esa situación y que hayan sido determinadas por el líder del proyecto.

Para componentes del tipo Necesidad, el contexto de mayor interés será posiblemente el de actividad. Recordemos que se había hablado que las necesidades son abstractas y que su contexto es el contexto de su fuente, entonces el contexto relacionado a las actividades que ejecuten estas fuentes sobre las necesidades serán las que se consideren o no como de interés para un proyecto.

Como ejemplo, evaluemos la situación hipotética de que un proyecto tiene una lista de requerimientos que sirven de base para el análisis y posterior diseño e implementación de una solución de software. Podríamos decir que la lista de requerimientos son las necesidades del proyecto, la persona asignada como responsable de estos documentos será a quién se le controle su actividad en relación a la gestión de las necesidades. Imaginemos que el responsable decide ejecutar cambios sobre la lista de requerimientos, estos cambios deberán reportarse por ejemplo al grupo de analistas funcionales, al líder de proyecto, etc. El sistema permitirá al administrador del proyecto indicar que cuando se modifiquen las necesidades se notifique al respecto a una serie de destinatarios, esta facultad evita que cambios realizados en las necesidades de un proyecto no sean informados a quien corresponda.

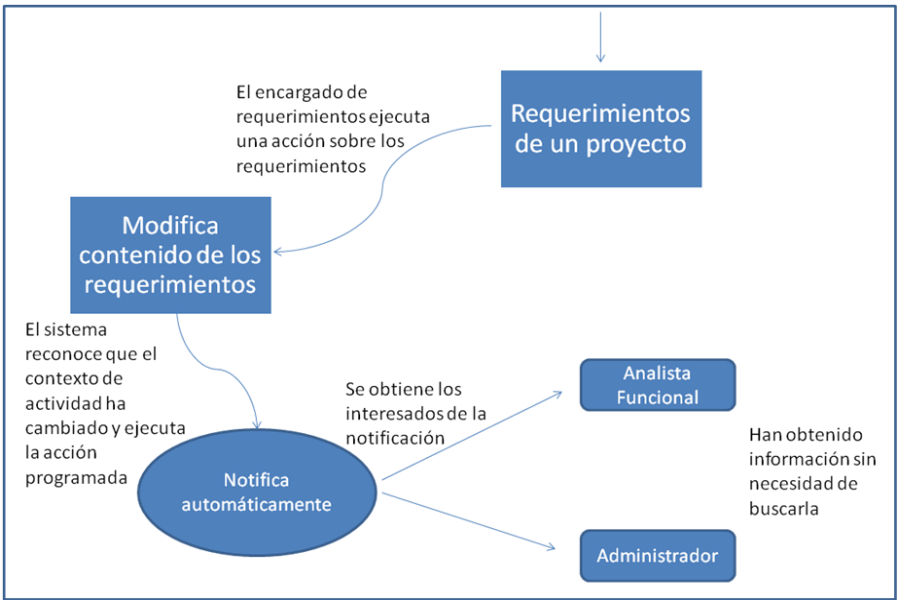


Ilustración 11: Efecto de un cambio contextual

En realidad las acciones a tomar ante la presencia de un determinado contexto podrían ser varias y también debemos tener presente que habrá contextos que no requieran la ejecución de ninguna acción (ya que no son de interés para el proyecto). Con la intención de simplificar: si un contexto no es de interés, pues no deberá ser considerado como un posible estado contextual al cual un elemento pueda llegar.

Para los componentes de tipo Actividad, los contextos de interés pueden variar también dependiendo de diversas circunstancias del proyecto. Pensemos en una actividad llamada entrega

de análisis, ésta actividad provee el insumo al grupo de diseñadores de software, por lo tanto su contexto relacionado a la actividad es de interés para los diseñadores, entonces sería conveniente configurar al sistema para que informe sobre la finalización del análisis a quienes esperan estos documentos como insumos.

El Producto es también un componente del proyecto, si un producto tiene problemas en el mercado, muy probablemente va a tener que ser repensado en busca de una mejor funcionalidad, en este sentido se podría considerar por ejemplo el contexto tiempo y definir que transcurrido un determinado periodo, se verifique, por ejemplo, el nivel de ventas, o la cantidad de reclamos receptados sobre un producto y que se informe a una u otra persona del equipo que creó el producto para que se puedan considerar correcciones que eviten los problemas detectados.

El modelo de clases presentado a continuación facilita el ingreso, primero, de los estados contextuales de interés para un componente del proyecto, así como la determinación de la o las acciones a tomar cuando un componente alcance un estado contextual específico.

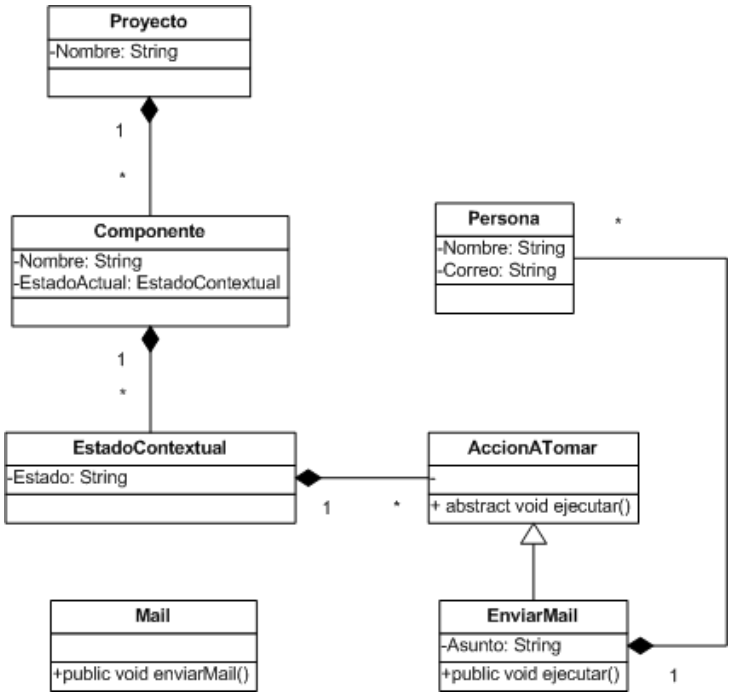


Diagrama 3: Definición de Estados Contextuales y Acciones a tomar

El modelo de clases del diagrama 3, agrega una clase llamada EstadoContextual, esta clase permite definir un conjunto de estados contextuales a los cuales un determinado componente de un proyecto puede llegar, en el ejemplo de la ilustración 10, las instancias de esta clase serían: creándome, modificándome y eliminándome.

La clase AccionATomar es abstracta y permite definir como subclases n acciones a tomar ante la ocurrencia de un contexto. En el modelo de clases se presenta como acción concreta a tomar, a EnviarMail, pudiéndose definir otras acciones concretas. La clase EnviarMail contiene una colección de objetos de la clase Persona que se especifican como destinatarios del mail, el asunto del correo se define en el atributo Asunto, el método ejecutar() se define para generar las actividades requeridas para que se efectivice el envío de correos a los destinatarios.

La clase Mail, conoce como realizar el envío del mail que la clase EnviarMail solicita.

En el ejemplo de la lista de requerimientos, recordemos que la intención era que cuando estos sean modificados se informe (a través de un correo) al analista funcional y al líder del proyecto, en ese caso el sistema manejaría las siguientes instancias:

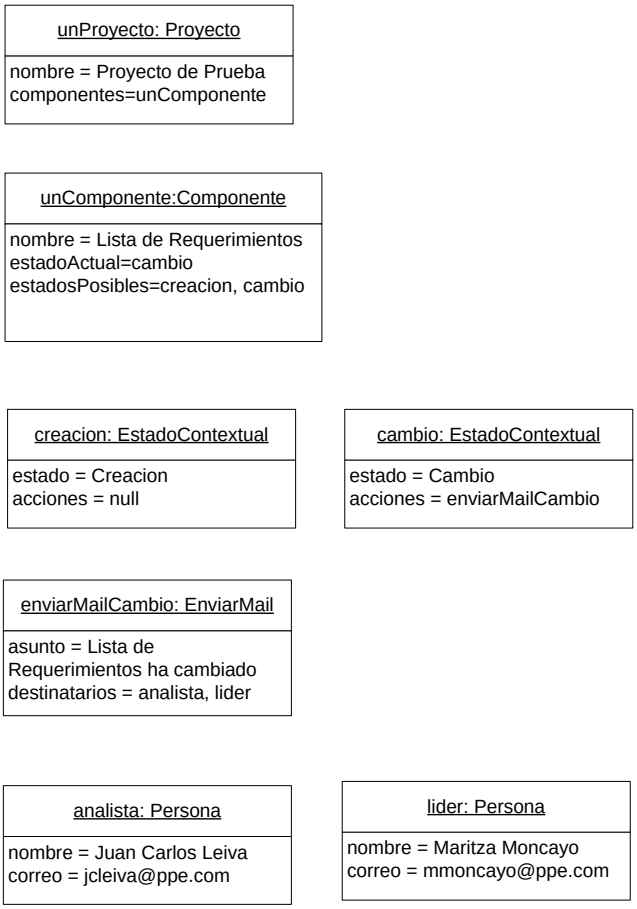


Diagrama 4: Instancias creadas ante un cambio contextual

En las instancias del diagrama anterior se puede ver que el proyecto tiene un único componente que a su vez tiene definido dos estados contextuales a los cuales puede llegar: Creación y Cambio. Para cada uno de los estados contextuales se define la acción o acciones a tomar, en el ejemplo, el estado contextual Creación no toma ninguna acción (por lo que inclusive, podría no ser creado) en tanto que el objeto Cambio define el envió de un correo electrónico por medio de la instancia enviaMailCambio, que conoce el asunto del correo y las personas destinatarias.

Cuando se hablaba de contextos de tipo actividad, decíamos que pueden existir algunas actividades a las cuales es interesante considerar su contexto, como en el caso de la actividad entregar análisis que cuando se finalizaba debía informar a quienes esperan a su producto como insumo, esta situación es aplicable en el esquema indicado en diagrama 3, ya que se puede configurar el envío de un mail a los interesados cuando el componente llegue al estado contextual “Finalización”.

Por otro lado están los componentes de tipo Producto, para este tipo se puede pensar en que a través de la computación ubicua se podría tener un sensor trabajando en verificar la curva de ventas del producto en distintos locales comerciales. El sensor informará cada cierto tiempo sobre el estado de la curva de ventas. En el diagrama de clases propuesto, se podría configurar al componente producto, con un estado contextual "Revisión de ventas", al cual llegará cada vez que su sensor notifique el estado de su curva y además será posible definir a quién informar respecto a estos cambios.

Es interesante destacar que el incluir al sistema en un entorno de computación ubicua permite obtener información de las distintas fuentes e integrarlas para beneficio del proyecto. De esta forma el proyecto deja de ser manejado de forma independiente del entorno donde se originó.

A través del conjunto de clases definidas en el diagrama 3 se permite al administrador de proyectos definir los componentes de su proyecto y para cada uno de ellos establecer un conjunto de estados contextuales a los cuales puede llegar y las acciones a tomar en cada caso. Si el administrador considerara que existen contextos que no son de importancia para el proyecto, entonces no los crearía como un posible estado contextual, de esta manera es posible definir los contextos de interés para un proyecto.

La cantidad de estados contextuales no tiene límite, en realidad son definidos por el administrador conforme sus requerimientos.

A manera de resumen podemos decir que se ha empleado un esquema: Componente - Estados Contextuales - Acciones a Tomar.

Con este esquema, el sistema podrá llegar a configurarse de cientos de formas distintas, siendo lo más importante la oportunidad de personalizar la funcionalidad que provee la Sensibilidad al Contexto para el proyecto actual.

Se puede notar además, que los sistemas de Administración de Proyectos, dejan de actuar de forma independiente a su contexto y más bien adaptan su funcionalidad conforme a lo que su realidad contextual indica.

1.26. Captando el contexto de interés

El atributo estadoActual de la clase Componente, indica en cuál de los posibles estados contextuales definidos se encuentra actualmente el componente, pero ¿cómo se actualiza este estado?

Una vez que se ha seleccionado y registrado el contexto de interés para un proyecto en particular, el siguiente paso a seguir será determinar la manera de captarlo para su futura utilización en los sistemas.

El contexto puede ser captado a través de sensores físicos ubicados en distintos lugares. Estos sensores son elementos capaces de vigilar constantemente un objeto y responder ante un cambio informando a alguien o a algo al respecto.

La metodología y mecanismos de empleo de estos sensores no son materia de este trabajo de tesis, por lo que el análisis partirá considerando que los sistemas reciben constantemente las señales que emanan estos sensores, en los casos en los que se lo requiera.

El contexto también puede ser captado a través de software, por ejemplo en la manipulación de documentos digitales, pudiendo determinar cuando han sido creados, modificados, eliminados, además de proveer información sobre quién lo hizo, en qué momento, etc., así estos programas de software actúan también como sensores o agentes.

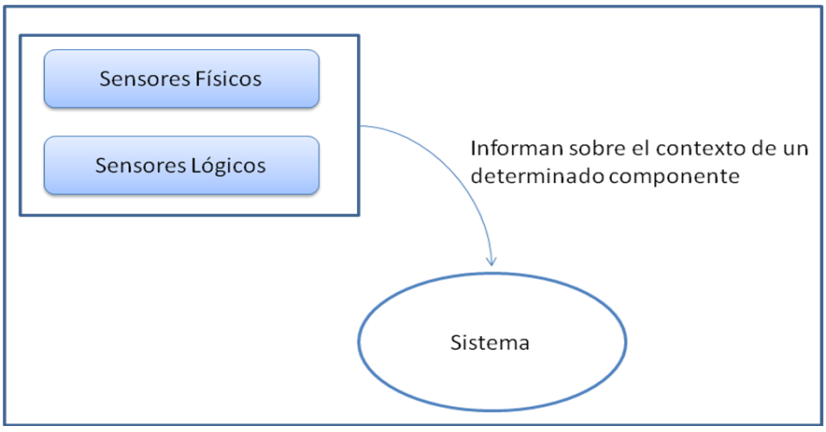


Ilustración 12: Funcionamiento de Sensores

De esta forma, los sistemas captan información desde los sensores, pero hay que diseñar un método para que dicha información sea transformada en algo útil para el desempeño del software.

Para que los sistemas puedan reconocer los cambios en el contexto de un componente (aquellos que los sensores o el software agente detectan e informan), será necesario dotar de un esquema de clases observadoras que ante un evento informado por un sensor ejecuten las acciones establecidas durante la configuración para cada estado contextual de los elementos de un proyecto.

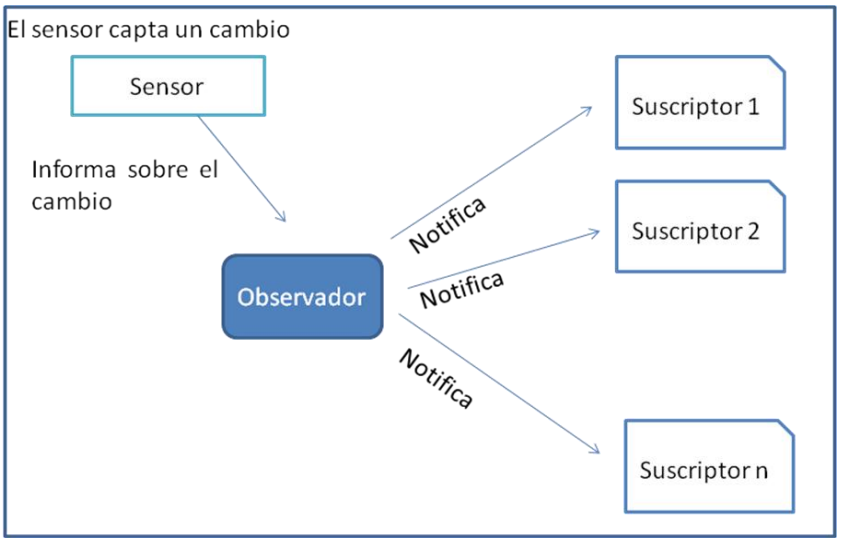


Ilustración 13: Un observador de los cambios contextuales

El patrón de diseño Observer ofrece un método a través del cual instaurar un conjunto de observadores sobre las clases que nos interesa conocer sus cambios contextuales. El patrón Observer presenta una clase Observable de la cual heredan los elementos que pueden tener un contexto de interés representados a través de la clase Componente**.** Cuando uno de los componentes sufra un cambio en su contexto de interés (es decir un evento ocurre), la clase Observable informará sobre el particular a la clase Observador**,** que será la encargada de gestionar la realización de la acción definida por el administrador del proyecto ante el cambio de contexto suscitado.

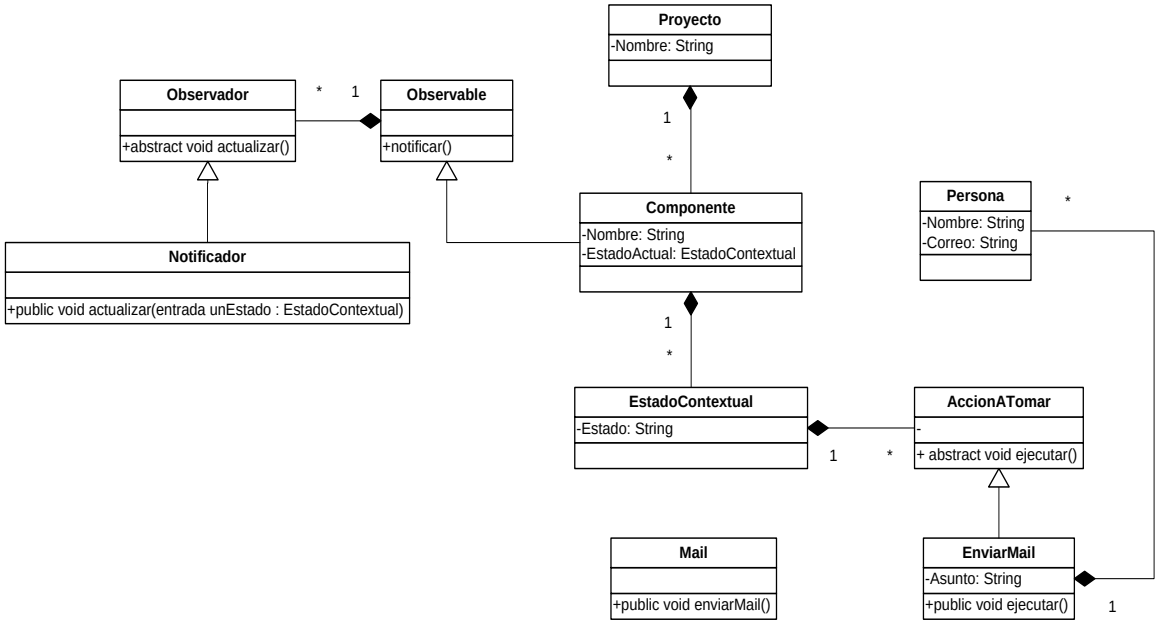


Diagrama 5: Observadores

La clase Notificador es el observador concreto, su trabajo es ejecutar las actualizaciones requeridas para que el estado del proyecto esté de acuerdo con el contexto actual de sus componentes.

La forma en que las clases colaboran para actualizar el estado del componente y ejecutar las acciones definidas se representa en el siguiente diagrama de colaboración:

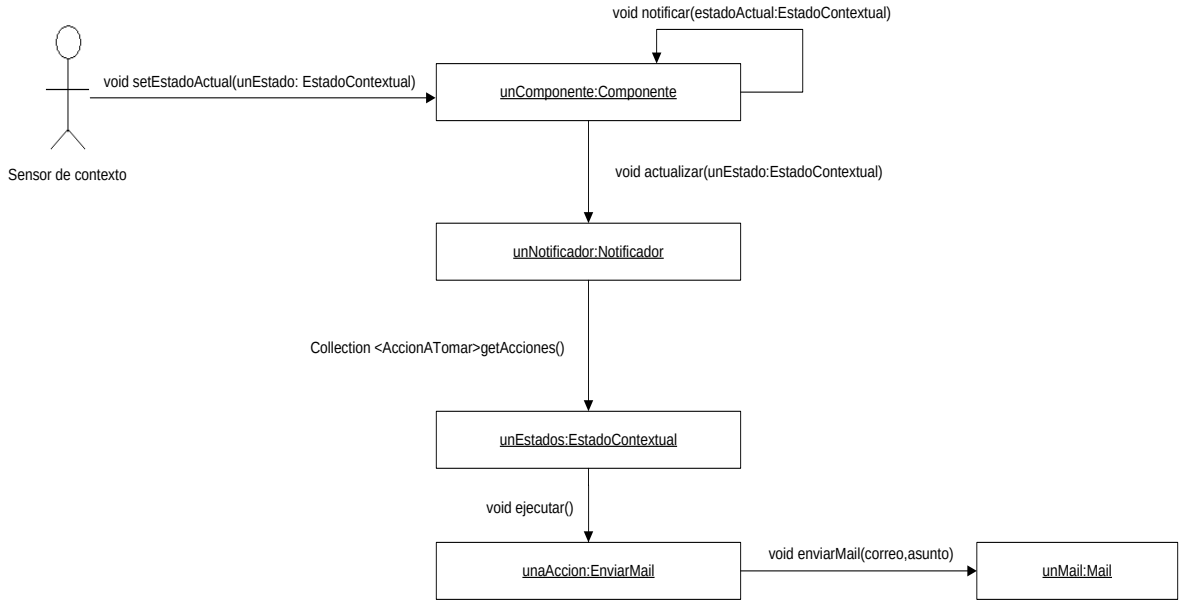


Diagrama 6: Actualización de un estado contextual de un Componente

1.27. Consiguiendo información de forma automática

El esquema de observadores planteado en los párrafos que anteceden, permite solventar uno de los aspectos detectados como limitantes de los sistemas para Administración de Proyectos: *la dependencia para obtener información.*

Los sistemas diseñados para la Administración de Proyectos, esperan que algún encargado pueda suministrar información referente al proyecto (en general, a los cambios de sus componentes) y no tienen las herramientas para obtenerla de forma autónoma.

Permitir que los aplicativos conozcan su contexto y las acciones a tomar ante eventos de importancia, proporcionan a los sistemas de Administración de Proyectos la oportunidad de conocer datos de forma automática. De esta manera, en la mayoría de las ocasiones los sistemas serán informantes antes que informados.

Al pasar la responsabilidad al sistema sobre la notificación de cambios en los componentes de un proyecto, se minimiza enormemente la carga de responsabilidad de los miembros de un equipo respecto a estos aspectos, además que al eliminar la posibilidad de ausencia de notificaciones, los

proyectos tienen grandes oportunidades de encaminarse correctamente a la consecución en tiempo y forma.

1.28. Otras acciones a tomar

En las secciones anteriores pudimos ver cómo a través del conjunto de clases definidas en el modelo se podía determinar acciones a tomar ante un contexto. La acción a tomar del ejemplo, se representó por medio de la clase EnviarMail, sin embargo es posible crear otras subclases de la clase AccionATomar, que realicen tareas distintas al envió de un mail.

Pensemos en el componente “Reunión de avance” de un proyecto. El presidente de la reunión podría registrarlo como parte del proyecto y seleccionaría la fecha, lugar para realizarla y por supuesto los asistentes. En esta situación se podría establecer como contexto de interés su creación, cambio y finalización.

En el estado contextual creación, las acciones a tomar podrían ser enviar un mail a los asistentes de la reunión (lo que es factible a través del uso de la clase EnviarMail), pero también va a ser de utilidad ingresar la reunión en la agenda de los asistentes, lo que no puede ser hecho con el método ejecutar() de la clase EnviarMail, en este caso se va a requerir una nueva subclase de la clase AccionATomar y por medio del polimorfismo, ejecutar la acción requerida.

En ese sentido se podría definir la clase Agendar, cuyo método ejecutar sepa como ingresar en las agendas electrónicas de los asistentes la reunión planificada.

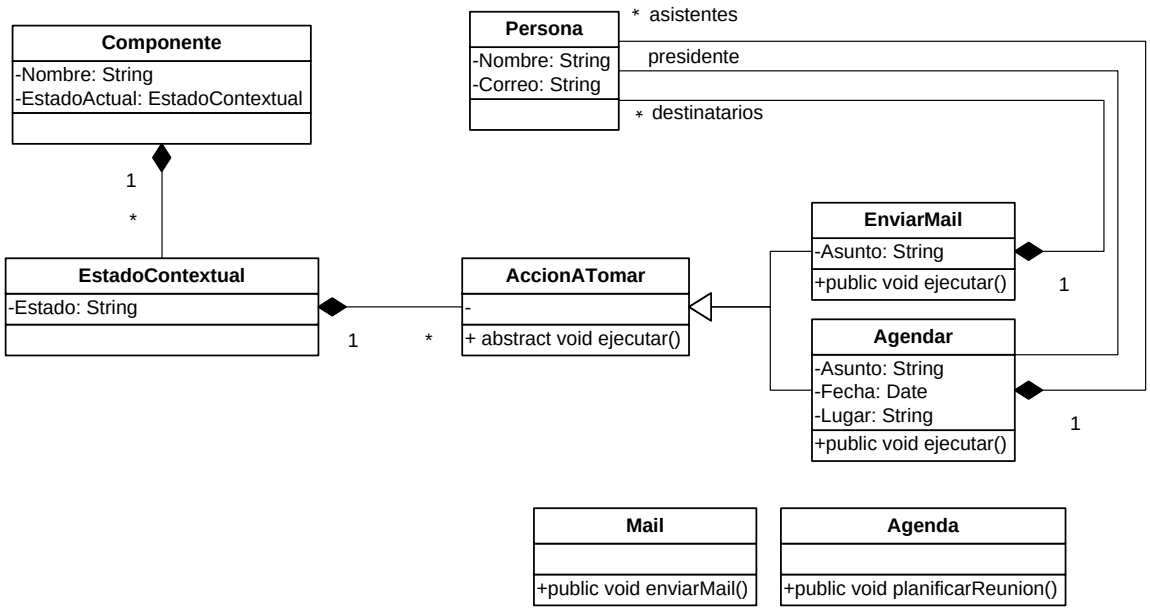


Diagrama 7: Clase Agendar

En el diagrama 7, es posible observar la subclase Agendar, que conoce el asunto de la reunión, sus asistentes, presidente, lugar y fecha de realización. Además el método ejecutar sabe que debe relacionarse con la clase Agenda para realizar la acción definida.

Lo interesante de este esquema es en primer lugar, la oportunidad de planificar reuniones en la misma herramienta de administración de proyectos, dejando a la clase Agenda el trabajo de ingresar efectivamente una reunión a la agenda de un participante. La agenda electrónica podría ser parte integrante del sistema o no, en caso de que se empleara una herramienta independiente para la agenda electrónica, la clase Agenda, debería en su método planificarReunión, establecer las interfaces requeridas con el sistema externo.

Automatizar el ingreso de reuniones en las agendas de los participantes ayuda a mejorar los niveles de comunicación en el proyecto, dado que sin importar si la herramienta de agenda electrónica está contenida en el sistema de administración de proyectos o es un software externo, se asegura que una vez que se registra la reunión como componente del proyecto, todos los interesados en la misma serán informados.

Hay que considerar además que se podría definir a más de la acción de agendar la reunión, acciones complementarias como por ejemplo enviar un mail para notificar el ingreso en la agenda, esto gracias a que la clase EstadoContextual define una colección de acciones a tomar.

Otra acción a tomar que se podría agregar al modelo, es una acción llamada Publicar, esta subclase de la clase AccionATomar, tiene el objetivo de publicar alguna noticia del proyecto en la página web de la empresa, cuando algún evento ocurre. Un ejemplo de aplicación de la clase Publicar puede ser la actividad: liberar nueva versión de un software, este acontecimiento puede ser de interés a todos los clientes que hayan adquirido el software y que tengan derecho a receptar las versiones nuevas que del producto se generen.

A través de la clase Publicar se podría indicar, en donde se publica, qué se publica y a qué página conduce la publicación.

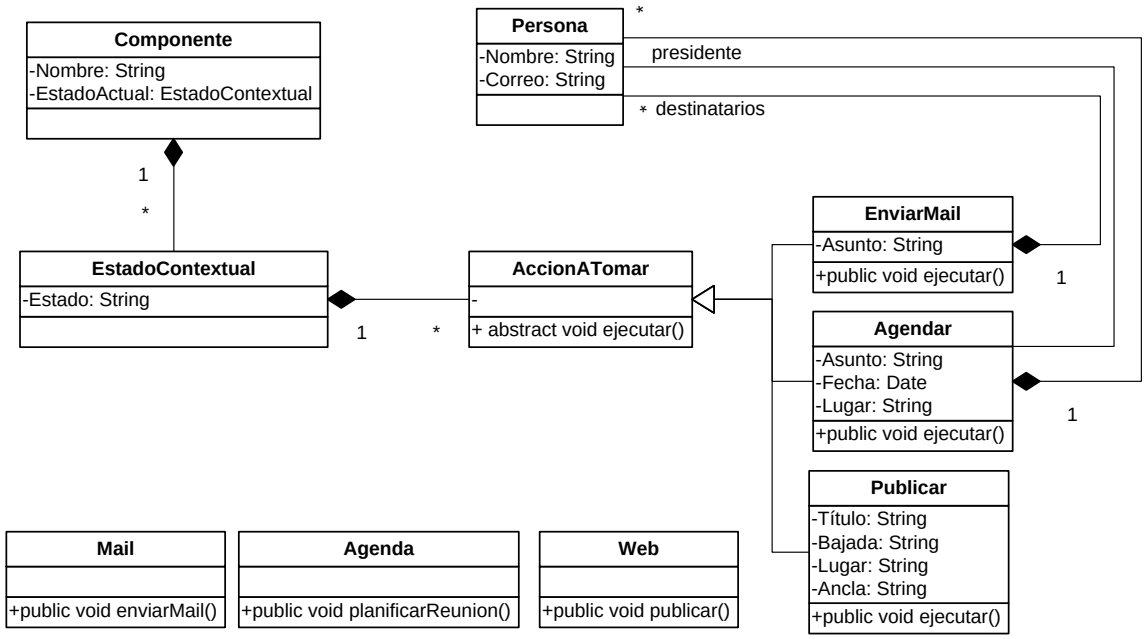


Diagrama 8: Clase Publicar

La clase Publicar en su método ejecutar() sabrá que por medio de la clase Web, podrá efectivizar la publicación. En el siguiente diagrama de objetos se puede ver como se instancian las clases para este caso:

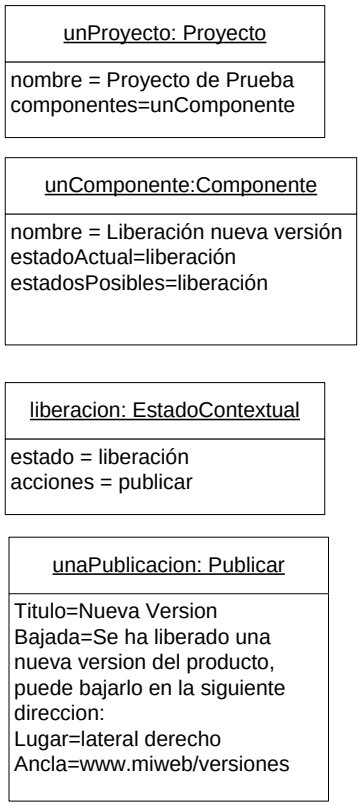


Diagrama 9: Instancias para la publicación

Las acciones a tomar pueden ser muy variadas y esto podría provocar una gran explosión de clases hijas de la clase AccionATomar. El modelo provee la oportunidad de especializar las acciones como en los casos indicados (EnviarMail, Agendar, Publicar), sin embargo sería útil contar con una solución genérica, que para los casos en que no se requiera una especialización, se puedan ejecutar acciones diversas.

Por esta razón el modelo sugiere el uso de programas externos, que ejecuten alguna tarea (la tarea para la cual fue desarrollado).

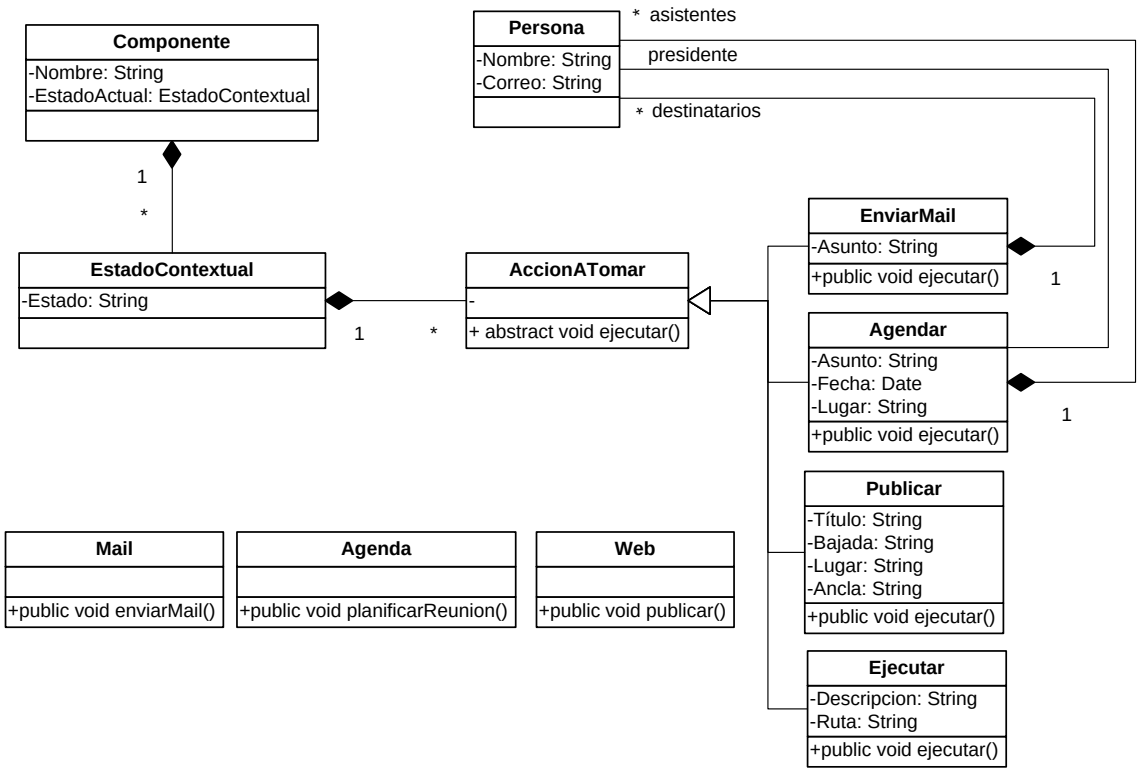


Diagrama 10: Clase Ejecutar

La clase llamada Ejecutar hija de la superclase AccionATomar, conocerá una ruta de ejecución de algún programa desarrollado para tomar una acción que no requiera ser especializada (sin parámetros) y en su método ejecutar, sabrá cómo realizar la ejecución del programa definido por la ruta. Si tuviéramos componentes de los proyectos que sean actividades cotidianas y que además se puedan compartir entre muchos proyectos, como por ejemplo actividades de control tal como revisar tareas con retrasos, se podría considerar programar un software que revise todas las tareas y verifique si aquellas que deben estar finalizadas lo están y si no es así que se emita un informe. Así el administrador del proyecto podría colocar actividades del proyecto (componentes) que sean para estos controles y configurar en la acción a tomar (acción ejecutar), la ejecución de un programa de software desarrollado.

En definitiva, el modelo de clases presentado ofrece muchas posibilidades para la definición de distintas acciones a tomar ante la ocurrencia de un contexto de uno de los componentes de un proyecto, será cuestión del equipo del proyecto y de su administrador una configuración adecuada para que las ventajas de la sensibilidad al contexto puedan ser de utilidad.

1.29. Usando el contexto para asistir a la toma de decisiones

Ser líder de un proyecto demanda la responsabilidad de tomar decisiones ante ciertos eventos y es obvio que decisiones acertadas van a producir mejores proyectos y el éxito a nivel profesional del líder.

Para tomar una buena decisión además de la experiencia o la intuición del líder, se debe considerar lo bien informado que el administrador del proyecto pueda estar. Un líder que conozca bien la situación de su proyecto, del cliente, del recurso asignado a la tarea o de circunstancias similares registradas en el pasado, podrá tener mayores argumentos a la hora de tomar una decisión acertada, sin embargo tener todo este panorama a la mano puede significar tiempo y con ello, costos.

El modelo del diagrama 5, debe ampliarse para capturar en sus instancias mayor información sobre el proyecto y su desarrollo para de esta manera proveer información útil a la hora de tomar decisiones. Fijémonos en el siguiente diagrama de clases:

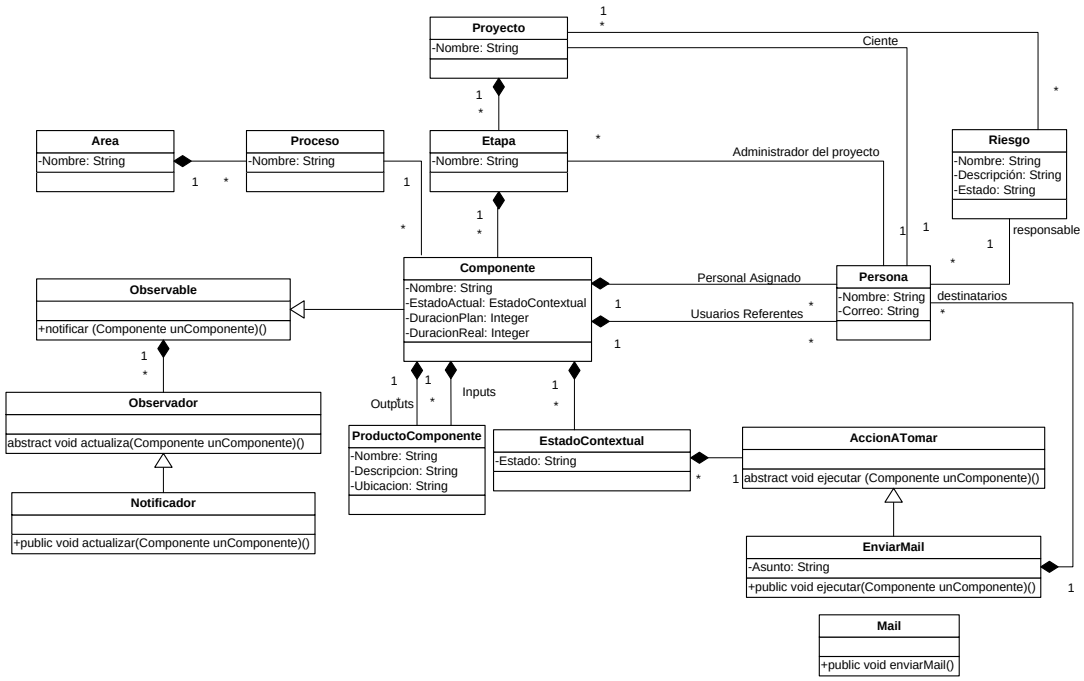


Diagrama 11: Incorporando mayor información

La clase Proyecto ya no posee una colección de objetos tipo Componente, sino más bien conoce todas las etapas en las cuales se ha dividido un proyecto, esto por medio de la asociación con la clase Etapa.

Cada Etapa conoce a los Componentes que la componen y cada Componente identifica al proceso del framework PMI al cual pertenece. Un área (áreas del framework PMI), posee una colección de procesos que la conforman.

También se ha visto conveniente hacer que la clase Componente identifique al conjunto de personas que son responsables de su ejecución y a los usuarios referentes que avalen la exactitud de la misma (de forma opcional). El uso de las asociaciones, Personal Asignado y Usuarios Referentes, permiten a la clase Componente conocer la colección de personas que desempeñan estos distintos roles y también, en el caso de personal asignado, facilita el uso del contexto identidad.

En las secciones anteriores se veía como contexto de interés a los contextos de tipo actividad y se procedía a definir un conjunto de acciones a tomar cuando cierta actividad era realizada, sin embargo no se explicitaba, quién o qué realizaba la actividad y por lo tanto era aplicable para todos los usuarios. Con la definición de un conjunto de personas responsables de las actividades, será posible verificar la identidad y la actividad para dar paso a la ejecución de acciones a tomar solo cuando una de las personas asignadas al componente estén ejecutando una actividad específica.

Todo Componente de un proyecto genera algo y en muchas ocasiones puede requerir algún insumo para iniciarse, ello justifica la creación de la clase de objetos ProductoComponente, las asociaciones Inputs y Outputs permiten que una instancia de componente conozca sus insumos y entregables.

Por otro lado, se ha agregado en la clase Componente más atributos para conocer mayor información sobre el componente, tal es el caso de DuracionReal.

Estos cambios y adiciones permiten conocer mayores detalles sobre el proyecto, además de definir algún tipo de clasificación de los componentes lo que será de utilidad más tarde. A continuación se presenta un extracto del diagrama de instancias en base a los cambios definidos.

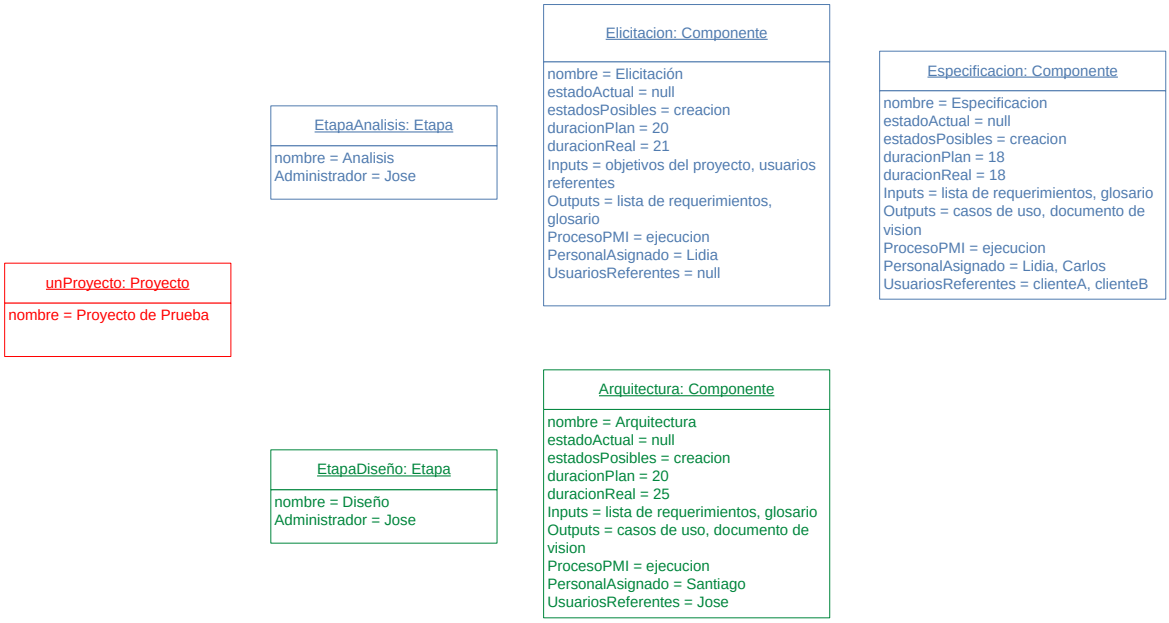


Diagrama 12: Instancias de un Proyecto

En realidad, la definición de las clases, sus atributos y asociaciones orientadas al registro de mayor cantidad de información respecto al proyecto puede ser la que se sugirió en la ilustración XX o podría implementarse de alguna otra manera, lo interesante se encuentra en poder identificar estos elementos para que luego a través de la sensibilidad al contexto sea posible proveer de datos útiles a la hora de tomar decisiones.

Para que el sistema opere con el objetivo de brindar apoyo a la toma de decisiones, es requerido manejar nuevamente el esquema: Componente - Estados Contextuales - Acciones a Tomar, visto en secciones anteriores. Cada ocasión que un componente llega a un estado contextual en el que se requiere la asistencia para la toma de decisiones el sistema ejecuta una o varias acciones a tomar definidas para el caso. Pueden existir muchas acciones a tomar que sean de utilidad para el análisis de una decisión, algunas de ellas se van a describir en los párrafos siguientes.

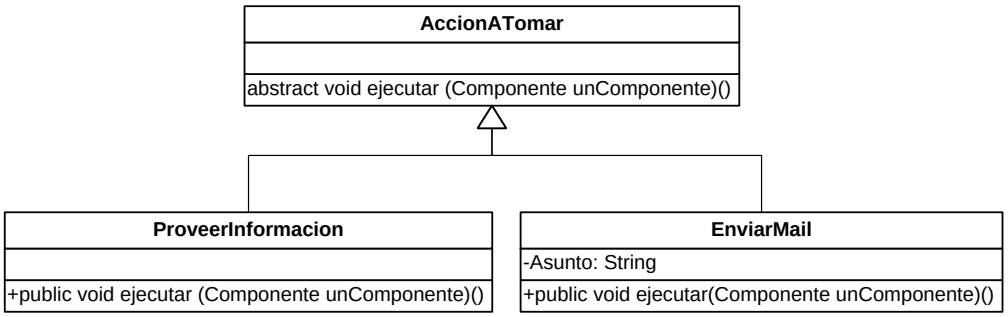


Diagrama 13: Clase ProveerInformación

La clase ProveerInformacion es una subclase de AccionATomar, y por tanto define un comportamiento distinto, por ejemplo, que el realizado por la clase EnviarMail.

La clase ProveerInformacion, podrá ser configurada para que se ejecute ante un contexto de interés determinado, en el que una de las acciones requeridas es que el sistema asista a la toma de decisiones por medio de la provisión de información relacionada a la actividad del componente actual, con la intención de que quién realice la consulta, posea gran cantidad de información de manera automática y muy relacionada al tema que está trabajando, de esta forma podría analizar los datos y tomar mejores decisiones. En el capítulo “Métodos para obtener información de interés”, se pueden ver detalles de cómo el sistema clasificaría la información a presentar y la pondría a disposición de los usuarios.

Otro interesante comportamiento, podría definirse a través del uso de una clase SugerirRiesgos hija de AcciónATomar, esta clase podría ejecutar un proceso que verifique los riesgos que están relacionados a proyectos, componentes o individuos similares en otros proyectos y podría presentarlos, por ejemplo, ante el estado contextual “Creación” del componente de proyecto: Lista de Riesgos.

La información contenida en las clases del modelo, más un adecuado conjunto de consultas facilitan la obtención de esta información. El creador de la lista de riesgos va a contar con datos que les serán de gran utilidad a la hora de tomar decisiones sobre su trabajo.

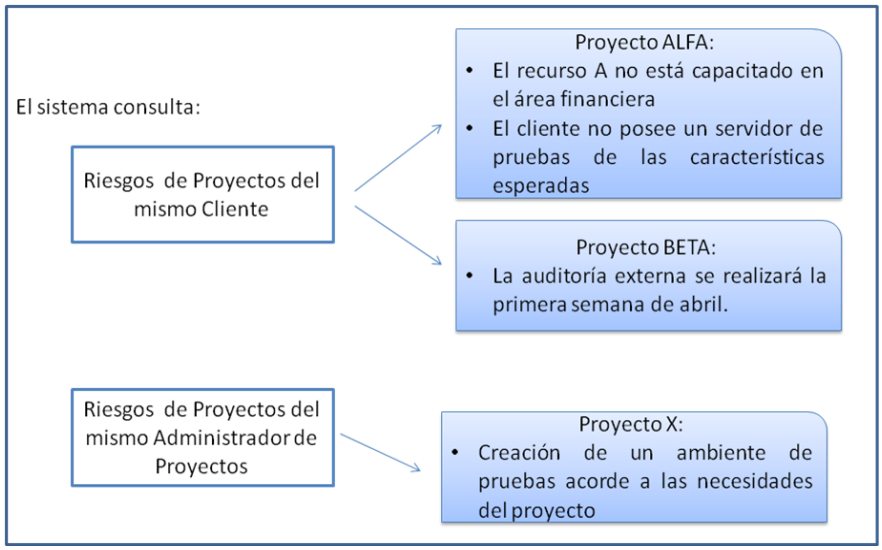


Ilustración 14: Datos relacionados a un riesgo

En el ejemplo de la ilustración 14, el interesado obtendría una lista de riesgos de varios proyectos. Estos riesgos, podrían ser analizados e inclusive incorporados como riesgos del proyecto actual. Si observamos, en el proyecto ALFA se cita un riesgo relacionado a la capacitación en temas financieros del recurso A, el líder del proyecto actual podría evaluar la factibilidad de incluir al recurso A en su proyecto aún sabiendo que tiene deficiencias en temas financieros, o podría indagar si el recurso obtuvo conocimiento en esa área que harían que el riesgo ya no exista. La bueno es que el líder del proyecto conoce temas que fueron relevados en el pasado y que podría no haber conocido sin la ayuda del software o que le hubiera tomado un trabajo de indagación obtenerlos.

La clase SugerirRiesgos, podría muy bien ser omitida y usarse una instancia de la clase Ejecutar, que conozca la ruta del proceso de consulta de riesgos.

Otra manera de ayudar a la toma de decisiones es sugerir recursos para el desarrollo de una determinada actividad. Como ejemplo consideremos que para una tarea determinada se sugiere asignar al recurso X. El sistema para hacer esta sugerencia, analizó que el recurso X, estará libre al momento que se requiera ejecutar la actividad en cuestión, también observó que el recurso X ha trabajado en proyectos similares o en proyectos con el mismo cliente o que el recurso X tiene una capacitación acorde a la actividad en cuestión. El sistema podría también, si fuera el caso, sugerir que el recurso X es el que más opciones tiene, pero que en el momento de ejecutar la actividad

estará de vacaciones y por tanto podría recomendar mover la actividad a otro momento o sugerir un recurso alternativo. El mecanismo a seguir es muy similar al descrito para el proceso de SugerirRiesgos.

Como se puede ver, la información y sugerencias que el sistema pueda ofrecer (en función no solo de información del proyecto actual sino haciendo uso de información de proyectos pasados y paralelos) es infinita.

Lo interesante es que con el esquema sensible al contexto: Componente - Estados Contextuales - Acciones a Tomar, es posible crear aplicaciones que puedan soportar esta funcionalidad.

1.30. Usando la información del pasado

Al incluir en el sistema a los productos de los componentes de un proyecto con trazabilidad (el modelo de clases nos permite conocer ascendientes y descendiente de un componente) y categorizados (en el capítulo “Métodos para obtener información de interés”, se habla de la aplicación de taxonomías ya sea usando folcksonomías u ontologías) al sistema de Administración de Proyectos, es posible indagar de forma automática para proveer datos de provecho para proyectos futuros.

La sección anterior mostró algunos ejemplos en los cuáles a través de la ocurrencia de un estado contextual para un componente de un proyecto es posible lanzar una serie de acciones encaminadas a proveer de datos útiles para la toma de decisiones de quien esté usando el sistema. Decisiones que pueden pasar por cuestiones tan complejas como determinar los riesgos de un proyecto hasta cómo saber que secciones tiene un determinado documento del proyecto.

La idea de integrar toda la documentación de un proyecto al sistema facilita que se encuentre disponible, para que al ser consultada, se presente a quien la requiera. El sistema de Administración de Proyectos, se convierte en el gran almacén de datos donde reside mucho del conocimiento de la empresa desarrolladora de proyectos y en la herramienta favorita de su equipo de trabajo.

Las ventajas innegables de la integración de la información presente y pasada, son el alto grado de reutilización de conocimiento de proyectos antiguos, la disminución de la cantidad de re-trabajo y la minora de los tiempos requeridos para hallar documentación de interés.

Métodos para obtener información de interés

1.31. Introducción

En el capítulo anterior se pudo ver que la provisión de información relacionada a una situación específica es una funcionalidad que disminuye la cantidad de trabajo requerido por parte de un recurso para obtener información sobre el proyecto e integra a los datos del pasado a los proyectos actuales consiguiendo hacer uso de la experiencia.

La acción a tomar ProveerInformación presentada en el diagrama 13, dispara un método que justamente ejecuta una serie de sentencias requeridas para conseguir información relacionada.

Cuando el contexto marcado como de interés sea alcanzado, el sistema deberá brindar la oportunidad de acceder a la documentación relacionada al tema actual, sin necesidad de buscarla sino más bien, el usuario la encontrará ya seleccionada en el sistema.

Contar con información que tiene alto grado de relación con el tema actual, facilita las decisiones que se deban tomar en torno al proyecto.

Para definir qué información puede llegar a ser de interés para un usuario en un momento determinado, existen varias alternativas como son el uso de folcksonomías y ontologías que proveen una manera de aplicar una taxonomía además de otras funcionalidades adicionales.

1.32. Usando folcksonomías

Las folcksonomías son etiquetas que los usuarios de la información consignan a los contenidos, para facilitar su ubicación futura, además de otorgarles, de alguna manera, una clasificación o taxonomía.

En la Administración de Proyectos existe una variedad de conceptos con los cuales se trabajan. Los miembros de un equipo de trabajo podrían consignar etiquetas a diversos componentes basándose en esos conceptos y en su percepción personal.

Los contenidos dentro del modelo clases que se planteó en el capítulo anterior, están representados a través de la clase ProductoComponente, y cada producto componente pertenece a una o varias instancias de la clase Componente.

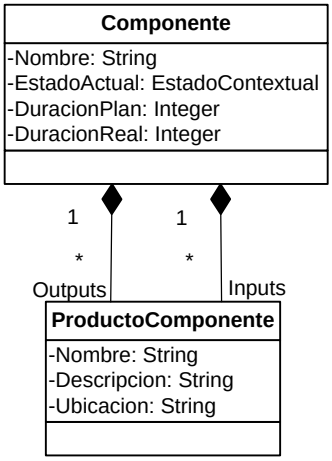


Diagrama 14: Clase ProductoComponente

Un componente tiene una colección de información que se le ha proporcionado como entrada así como productos de su gestión (los outputs). Por ejemplo, imaginemos la actividad del componente "Implementar Caso de Uso Crear Cuenta", este componente podría tener como inputs el documento de caso de uso, el documento de realización del caso de uso, el cronograma, etc. Su resultado u output sería un programa para crear cuentas, basado en el caso de uso Crear Cuenta.

Cada usuario de la información representada por la clase ProductoComponente, tendría la capacidad de etiquetar un contenido según su punto de vista.

Considerando el ejemplo anterior, podríamos pensar que el usuario desarrollador del proyecto, pudo marcar al documento de caso de uso con la etiqueta "Requerimientos de mi proyecto", además para la realización del caso de uso pudo haber usado la misma etiqueta. Por otro lado el analista funcional, pudo haber etiquetado al documento de caso de uso con la etiqueta "Casos de Uso del proyecto del cliente A", etiqueta que usó además para nombrar a los documentos "Caso de Uso para Modificar una Cuenta", "Caso de Uso para Eliminar una Cuenta". El analista funcional, podría buscar documentos relacionados a la etiqueta: requerimientos de mi proyecto que es de propiedad del desarrollador, sin embargo no podría modificar los documentos que tienen esa etiqueta ni podría agregar ni retirar documentos.

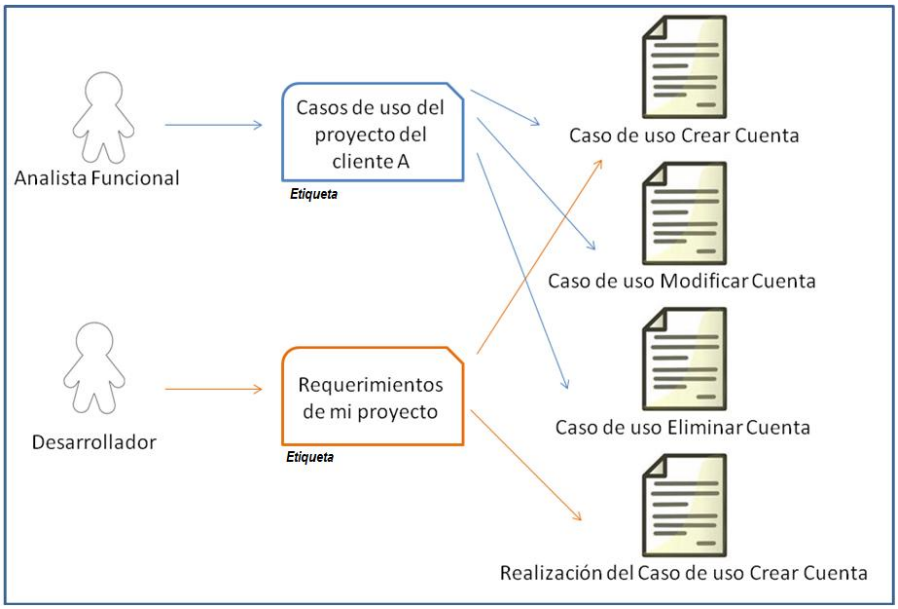


Ilustración 15: Etiquetas consignadas a documentos

La cantidad de etiquetas que se pueda consignar a un contenido es ilimitado, un usuario podría otorgar varias etiquetas a un mismo contenido y un contenido podría tener varias etiquetas consignadas por distintos usuarios.

Para permitir la tarea de tagging o etiquetado es requerido realizar modificaciones al modelo de clases propuesto hasta ahora, resultando lo siguiente:

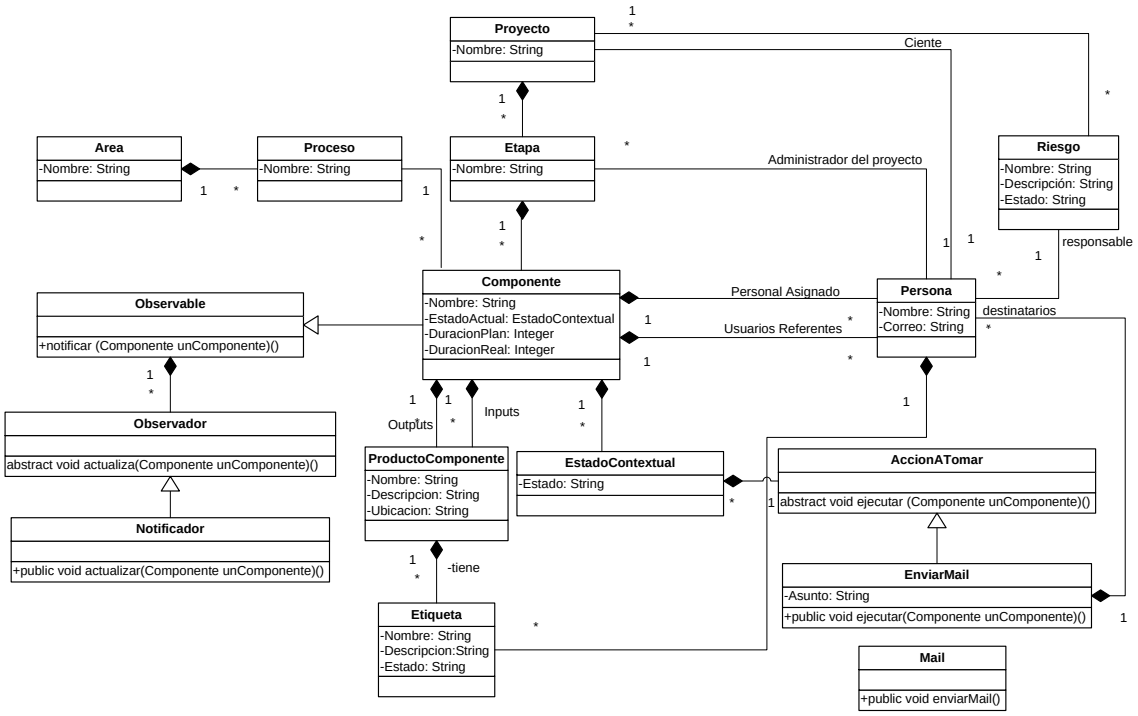


Diagrama 15: Clases para el empleo de Folcksonomías

Se ha definido la clase Etiqueta (o Tag), que contiene como atributos un nombre para la etiqueta y una descripción que detalle el ámbito de la etiqueta. En el ejemplo, la etiqueta con nombre "Requerimientos de mi proyecto", podría tener la siguiente descripción: "Documentos enviados desde el área de Análisis funcional para la elaboración del proyecto del cliente A, el 24 de enero de este año.".

El atributo estado, se orienta a indicar si la etiqueta esta activa o no, pues los propietarios de una etiqueta podrían decidir darla de baja, con lo que los documentos etiquetados con la misma perderían su referencia.

La asociación entre las clases "ProductoComponente" y "Etiqueta", identifican al conjunto de etiquetas que un usuario ha consignado a una instancia de la clase ProductoComponente. La asociación con la clase Persona, permite reconocer al propietario de una etiqueta.

Las etiquetas serán posibles consignarlas a nivel de ProductoComponente y no de Componente, pues cada input y output puede tener un criterio de clasificación específico.

El tagging estará disponible para todos los usuarios en todos los contenidos a los cuales tengan acceso.

Esa tarea de etiquetado dirigido por las personas que conforman el equipo de trabajo del proyecto, provee una oportunidad de clasificación de contenidos, misma que puede ser útil a la hora de traer a colación contenidos de interés para una actividad en particular.

Un usuario no podría usar una etiqueta de otro usuario para marcar sus contenidos, pero sí podrá usarla para consultar información.

Cuando se alcance un estado contextual que requiera la provisión de información de interés, el sistema indagará por todas las etiquetas que se han otorgado al contenido en cuestión y para cada una de las etiquetas creará un enlace a todos los demás contenidos etiquetados con la misma etiqueta, esto, gracias al criterio de clasificación de los usuarios, proveerá información relacionada al tema, que bien servirá de soporte para la toma de decisiones y la mejora continua en el proyecto.

Retomando el ejemplo, cuando un usuario ingresa al contenido “Caso de Uso Crear Cuenta”, el sistema pondrá a disposición las etiquetas: “Casos de Uso del proyecto del cliente A” y “Requerimientos para mi proyecto”. Si el usuario accede al enlace de la primera etiqueta, podrá obtener información sobre otros casos de uso relacionados, como casos de uso de creación y eliminación de cuentas. Ese sería solo un primer nivel de acceso, y probablemente el que mayor grado de relación de los contenidos provea, sin embargo sería posible explorar en otros niveles.

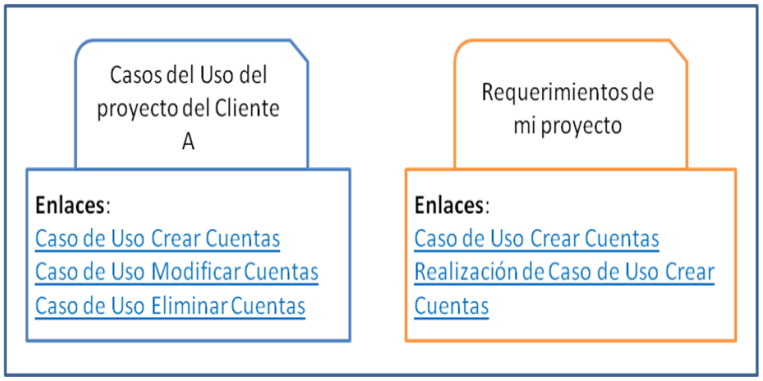


Ilustración 16: Información Relacionada por medio de Etiquetas

Puede ocurrir, que la información alcanzable a través del uso de folcksonomías, termine permitiendo el acceso a documentación, datos, programas, etc, a los cuales ciertas personas no están autorizadas, en ese sentido es requerida la implementación de un sistema de autorización para los ProductoComponente del modelo de clases propuesto. La solución se centra en la creación de una relación entre las clases ProductoComponente y Persona, llamada lectura. El sistema por defecto permitiría la lectura de los ProductoComponente a sus creadores, y estos si desean pueden compartirlos con otros miembros del equipo de trabajo, de esta manera se resguarda la confidencialidad de la información.

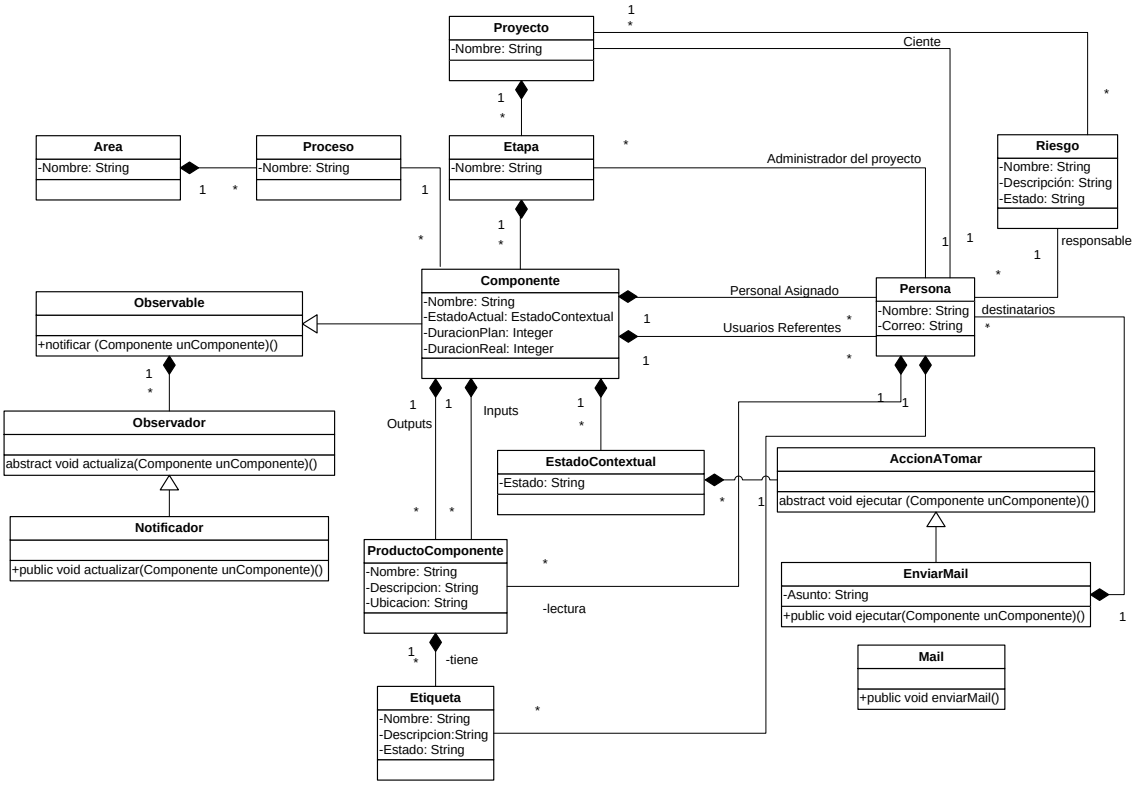


Diagrama 16: Relación Lectura

Cuando se ingrese a un enlace que la folcksonomía provee como de interés para un usuario, el sistema presentará el ProductoComponente relacionado al enlace y nuevamente indagará sobre las etiquetas asignadas al mismo, de esta forma se puede profundizar en niveles de relación entre documentos. En el ejemplo de la ilustración 15, supongamos la situación en la que un usuario está ubicado en el documento "Caso de uso Modificar Cuenta" el sistema presentará como información de interés a todos los documentos marcados con la misma etiqueta, tal es el caso de

"Caso de uso Crear Cuenta" y "Caso de uso Eliminar Cuenta", luego el usuario podría acceder al documento "Caso de uso Crear Cuenta" y cuando el sistema lo presente, nuevamente mostrará una lista de documentos relacionados ahora al documento visitado actualmente, en este sentido mostrará "Caso de uso Crear Cuenta" y "Caso de uso Modificar Cuenta" que comparten su etiqueta "Casos de uso del cliente A", pero también se presentará enlaces a "Realización de Caso de uso Crear Cuenta", que comparte la etiqueta "Requerimientos de mi proyecto".

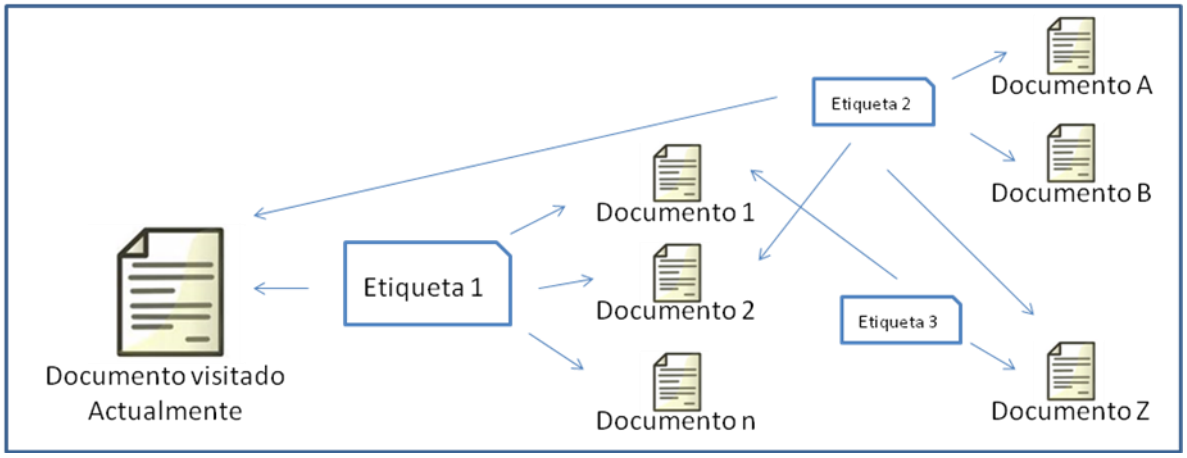


Ilustración 17: Conexiones a través de etiquetas Folcksonómicas

La ilustración 17, muestra cómo las etiquetas de las folcksonomías generan muchas conexiones Etiqueta – Documento que relacionan información sin una estructura específica sino exclusivamente basándose en el criterio de los usuarios. Con ello se puede concluir que no existe un número de niveles máximo que pueda tener esta categorización y además que ciertos documentos pueden aparecer varias veces en el transcurso de una indagación de una ruta.

Si bien el etiquetado representa una muy buena alternativa de clasificación de contenidos, hay siempre que tener en la conciencia que las folcksonomías son apreciaciones de cada una de las personas involucradas en un proyecto, y que por tanto pueden, como no, representar el verdadero significado de la documentación. Esta es una de las desventajas de este método, su falta de formalidad, lo que podría ocasionar de cierta manera que la taxonomía aplicada no sea la correcta para cada uno de los casos.

1.33. Usando ontologías

Las ontologías, son definiciones formales de algún concepto que además son compartidas por todo el equipo de trabajo de un proyecto, circunstancia que les brinda mayor robustez semántica.

En la sección anterior se pudo observar como por medio de una clasificación taxonómica fue posible definir contenidos relacionados a un documento en especial, sin embargo se anotó que la clasificación taxonómica que emplea folcksonomías (a pesar de ser un mecanismo útil y muy cercano al usuario) carecía de formalidad dado su origen en el conocimiento individual de quienes conforman un proyecto.

Con el uso de las ontologías se puede otorgar formalidad a esa clasificación así como la oportunidad de aplicar razonamientos basados en los conceptos existentes y sus propiedades. Por ello, se considera el uso de ontologías como un mecanismo alternativo para definir un conjunto de información de interés a un tema dado en el entorno de la Administración de Proyectos.

Las ontologías, a diferencia de las taxonomías, no se dedican exclusivamente a marcar contenidos (es decir a clasificarlos), sino en adición otorgan relaciones entre los conceptos existentes (propiedades), cuestión que provee la oportunidad de razonar sobre ellos.

Una buena práctica a todo nivel en el desarrollo de un proyecto, es justamente, hablar un lenguaje estándar, que permita definir sin ambigüedad lo que un término significa en su dominio.

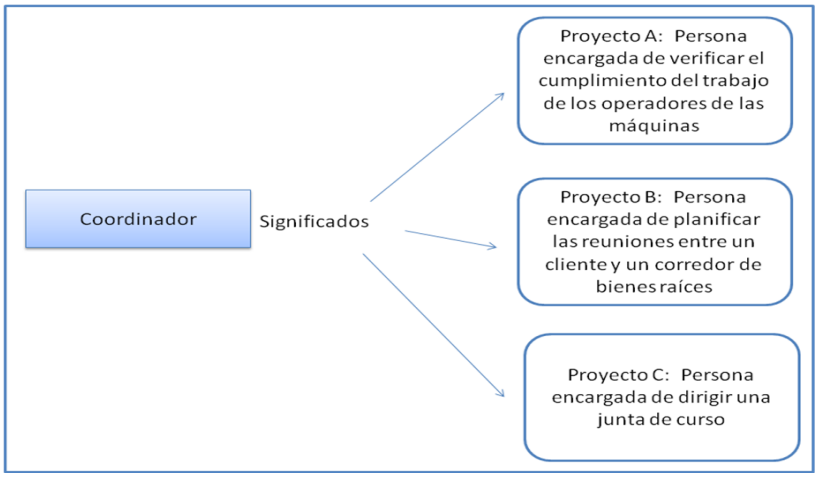


Ilustración 18: Significado de un término según el dominio

La palabra Coordinador, por ejemplo, puede tener diversos significados en función del dominio en el cuál se esté usando, tal es el caso del Proyecto A, en el cual un coordinador se dedica a labores de verificación del cumplimiento del trabajo de otros empleados, en cambio en el Proyecto C, un coordinador es la figura que dirige una junta de curso. Con estas convenciones en mente los miembros del equipo del Proyecto C, al pensar en un coordinador se estarán imaginando a quien convoca y dirige una junta de curso y no al encargado de supervisar el trabajo de unos operarios. En definitiva, cada palabra puede tener un significado distinto en función del dominio en el cual se desenvuelve. Lo interesante está en llegar a definirlo como una convención aceptada por todos quienes trabajan en el dominio.

Las ontologías permiten especificar estos conceptos con el uso de una tripleta sujeto – predicado – objeto que puede ser escrita con alguno de los lenguajes ontológicos citados en el capítulo Ontologías.

Con la intención de que las ontologías pongan a disposición de nuestros fines su funcionalidad, se requiere la definición de un conjunto de conceptos que vayan a ser aceptados dentro de una organización encargada de la ejecución de proyectos. La ontología a emplearse debería centrarse en la definición de los conceptos requeridos en el dominio de la Administración de Proyectos.

Esta labor es por demás compleja, debido a que la especificación de conceptos y relaciones entre ellos depende de un alto grado de conocimiento en el dominio, en este caso, de la Administración de Proyectos. Para conseguir una ontología va a ser necesario el concurso de expertos en el tema que puedan generar una ontología válida para el manejo de proyectos. Orientados a ese objetivo existen varios trabajos que pretenden crear ontologías para la Gestión de Proyectos [22].

Para el presente trabajo de tesis, se va a usar una ontología elemental sobre algunos conceptos de la Administración de Proyectos, misma que puede ser vista en el Anexo A “Ontología de Prueba” y que no representa una ontología aplicable en la realidad, su razón de ser es: servir para determinar un método a través del cual se pueda probar la eficiencia de su uso para seleccionar un conjunto de información relacionada a un tema en particular.

Tal como en el caso de las folcksonomías, se requiere de un proceso a través del cual alguien o algo especifiquen la clasificación para un elemento determinado, en el caso de las ontologías, cada ProductoComponente deberá contar con una clasificación ontológica, o dicho de otra manera se le

deberá clasificar como un concepto definido en la ontología aplicada. Esta operación sería efectuada por su creador y aunque sería beneficioso, no es un proceso obligatorio (como tampoco lo era el clasificado folcksonómico). El creador de un ProductoComponente, en realidad no va a seleccionar una etiqueta ontológica para el elemento, lo que va a ser es describir al elemento en términos de sus propiedades.

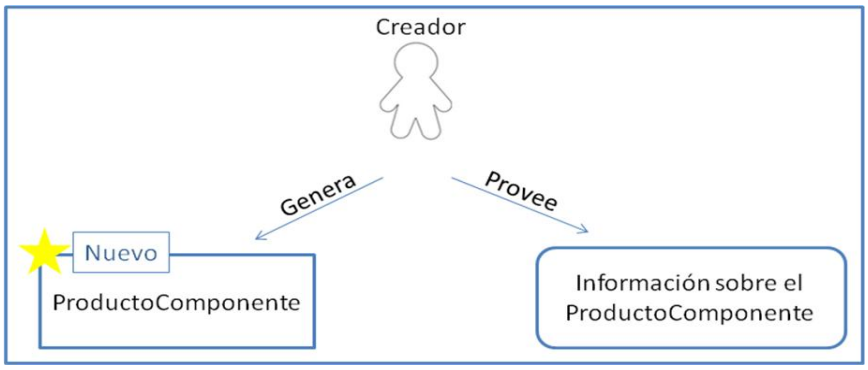


Ilustración 19: Entrega de información sobre un ProductoComponente

Luego, la ontología podrá iniciar un proceso de inferencia que determine la mejor clasificación para el elemento en cuestión, conforme las propiedades descritas por su creador.

Imaginemos por un momento, que en lugar de realizarse el proceso anterior, el creador de un ProductoComponente tuviera que seleccionar un concepto adecuado para el documento con el cual está trabajando. En primer lugar, el creador debería poseer conocimiento exhaustivo sobre la conformación de la ontología (sus estructuras y jerarquías) y el significado de cada uno de sus conceptos, para de una manera inequívoca colocar al elemento en un concepto adecuado. Por otro lado, no habría manera de asegurar que la selección realizada sea la correcta, lo que podría redundar en inconsistencias a la hora de probar nuestra ontología.

Por lo anotado, es más fácil y aplicable el hecho de que el creador le "cuente" a la ontología sobre el elemento con el que está trabajando y le delegue a la ontología el trabajo de clasificarlo, sabiendo que la ontología conoce a plenitud todos los significados y jerarquías y podrá indagarse a sí mismo (por medio de la inferencia), cuál de sus conceptos es el que mejor calza.

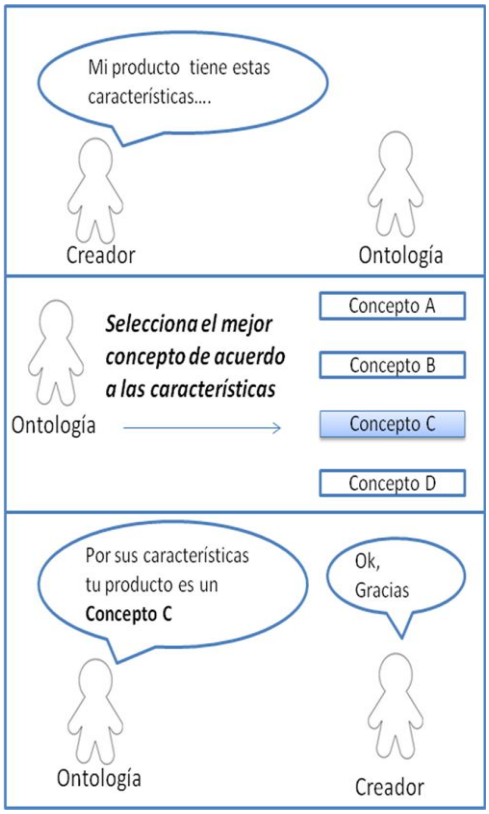


Ilustración 20: Proceso para determinar el concepto ontológico adecuado

Un caso de uso es un artefacto empleado ampliamente en la definición de requerimientos para proyectos de desarrollo de software, es una manera de especificar todos aquellos deseos, necesidades y expectativas que tienen los usuarios respecto a un proyecto. Cuando se posee el conjunto de todos los casos de uso para un software, es posible presentarlos a una audiencia para que los valide y apruebe. Muy probablemente, un ProductoComponente del modelo de clases propuesto va a hacer un caso de uso.

Podemos usar este ejemplo para conceptualizar por medio de una ontología a un caso de uso. La clase podría llamarse CasosDeUso, después se procede a dar propiedades tales como decer que es un documento (de aquellos producidos durante la etapa de análisis de un software), que fue escrito por alguien cuyo perfil en el proyecto es de Analista Funcional o Ingeniero de Requerimientos, también se podría decir que un caso de uso existe para solventar uno o varios requerimientos funcionales del proyecto.

El siguiente, es un extracto escrito con OWL DL para definir las características identificadas para un caso de uso:

```
<owl:Class rdf:about="CasosDeUso">
<rdfs:subClassOf rdf:resource="Documentos_de_Analisis"/>
<owl:ObjectProperty
                    rdf:about="TieneReferenciaAElemento"/>
</owl:onProperty>
<owl:someValuesFrom>
<owl:Class rdf:about="Requerimiento_Funcional"/>
</owl:someValuesFrom>
<owl:onProperty>
<owl:ObjectProperty rdf:about="TieneAutor"/>
</owl:onProperty>
<owl:someValuesFrom>
<owl:Class rdf:about="Analista"/>
   </owl:someValuesFrom>
</owl:Class>
```

OWL, define el concepto para un Caso de Uso creando una clase ontológica llamada CasosDeUso, también describe las propiedades en referencia a otros conceptos, como en el caso de la propiedad TieneAutor que se refiere a instancias de la clase Analista. CasosDeUso, es una subclase de una clase superior llamada Documentos_de_Análisis y finalmente está la propiedad TieneReferenciaAElemento que permite indicar el requerimiento funcional al que un CasoDeUso da solución.

De manera similar, es posible establecer el concepto de otros elementos de un dominio, basándose en su significado propio y por las propiedades que en torno a otros conceptos mantiene.

Supongamos la existencia de los siguientes documentos en un proyecto para una aerolínea:

- Documento A. Creado por Jorge. Soluciona el Requerimiento Funcional de mantener el registro actualizado de pasajeros.
- Documento B. Creado por Antonio. Diseña el diagrama de secuencia para el Documento A así como define el correspondiente conjunto de tablas para su persistencia.

- Documento C. Creado por Jorge. Soluciona el Requerimiento Funcional de vender boletos para vuelos.
- Documento D. Creado por Antonio. Soluciona el Requerimiento Funcional de mantener información acerca de los sobrecargos de un vuelo.

Como en la ilustración 20, los creadores de los documentos A, B, C y D, comentarán sus características a la ontología, la misma que procederá a analizarlas, lo primero que hace es establecer los valores para cada propiedad de un documento:

Nombre	Es un documento	TieneReferenciaAElemento	TieneAutor
Documento A	Si	Si. A un Requerimiento Funcional.	Si. Un Analista Funcional.
Documento B	Si	No. Se refiere a un Caso de Uso	Si. Un Arquitecto de Software
Documento C	Si	Si. A un Requerimiento Funcional.	Si. Un Analista Funcional.
Documento D	Si	Si. A un Requerimiento Funcional	Si. Un Arquitecto de Software.

Tabla 4: Propiedades definidas para unos documentos

Con eso listo, procede a evaluar el concepto de CasosDeUso, es decir, verifica que sea un Documento, que se refiera a un Requerimiento Funcional y finalmente, que su autor sea un Analista Funcional. Solo si cumple con esas características entonces se podría decir que el elemento en cuestión es un caso de uso.

Para el Documento A se procede de la siguiente manera: Sí es un documento, sí se refiere a un Requerimiento Funcional y sí tiene un autor cuyo perfil es Analista funcional. La ontología indica entonces que el Documento A es un CasoDeUso ya que cumple con todas las características para serlo.

Para el Documento B se procede así: Sí es un documento, no se refiere a un Requerimiento Funcional y no tiene un autor cuyo perfil es Analista funcional. La ontología indica entonces que el Documento B no puede ser un CasoDeUso.

Se procede a trabajar con el Documento C: Sí es un documento, sí se refiere a un Requerimiento Funcional y sí tiene un autor cuyo perfil es Analista funcional. La ontología indica entonces que el Documento C es un CasoDeUso.

Para terminar, consideremos al Documento D: Sí es un documento, sí se refiere a un Requerimiento Funcional y no tiene un autor cuyo perfil es Analista funcional. La ontología indica entonces que el Documento D no puede ser un CasoDeUso.

En los casos de los documentos A y D, se pudo definir la clase ontológica para el elemento analizado, sin embargo no se corrió con la misma suerte para los documentos B y D, ya que no cumplían con todas las condiciones definidas para que se los pueda catalogar como casos de uso. En estas circunstancias, lo más seguro será que exista una clase ontológica que defina a esos documentos de forma precisa.

Para el Documento B podríamos pensar en una clase ontológica llamada RealizaciónDeCasoDeUso, que sea un documento cuyo autor es un Arquitecto de software y que se refiera a un caso de uso. Si esta clase hubiera existido durante la evaluación que realizó la ontología, el Documento B caería en la clase RealizaciónDeCasoDeUso.

Ahora bien, aún teniendo esta última clase en la ontología, el Documento D, no hallaría su clase ontológica adecuada ya que no es un CasosDeUso porque su autor es un Arquitecto de Software y tampoco sería un RealizaciónDeCasoDeUso, debido a que se refiere a un Requerimiento Funcional y no a un Caso de Uso.

En este sentido podemos crear una clase ontológica que satisfaga ese concepto siempre y cuando sea un concepto válido en nuestro dominio. Al parecer, el ejemplo del Documento D, no es un concepto muy real, ya que a pesar de estar trabajando con elementos relacionados al análisis de un software (con son requerimientos funcionales), tiene como autor a un Arquitecto de Software que suele trabajar en etapas de diseño. En fin, la decisión de incluir o no un concepto a una ontología, resulta de un análisis detenido de su existencia en el dominio por parte de gente que conoce del tema a profundidad.

Regresando al modelo que se propone, éste debería permitir justamente que exista un paso de las característica de un ProductoComponente a la ontología para que se establezca la clase ontológica que le corresponde.

Para conseguir esa funcionalidad se hace necesario el uso de individuos de una clase ontológica. La relación existente entre un individuo y una clase ontológica, es muy similar al concepto del

paradigma Orientado a Objetos para clase y objeto: un individuo es una instancia de una clase ontológica.

El modelo soportaría la existencia de una ontología que describa todos los posibles conceptos de la Administración de Proyectos, luego, los ProductoComponente, se describirían y la ontología sería capaz de definir su clase ontológica adecuada. Con esa intención es imprescindible contar con una clase ontológica que englobe a todos los posibles tipos de ProductoComponente, de tal suerte que podamos crear instancias de esta clase superior y luego dejar a los razonadores de la ontología el trabajo de clasificar al individuo conforme la clase que más le convenga. Esta clase ontológica deberá contar entre sus propiedades a todas aquellas que cualquiera de sus clases hijas pueda tener, así se podrá crear un individuo de forma genérica. A esa clase ontológica le damos el nombre de MiDocumento.

La ontología como tal no cambiará en lo absoluto, ya que sus conceptos y propiedades vendrán definidas de antemano y aprobadas como conceptos válidos para el dominio. Sin embargo la cantidad de instancias irá en aumento conforme se vayan presentando los ProductoComponente a la ontología.

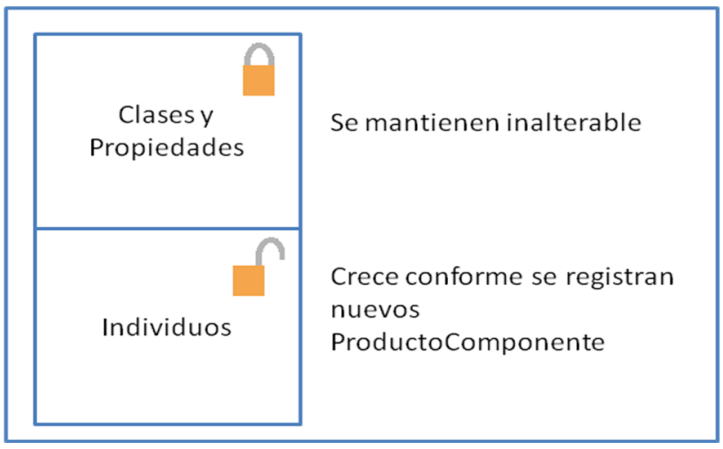


Ilustración 21: Esquema de crecimiento para la ontología a aplicar

El uso de las ontologías ha significado también la producción de motores de inferencia, capaces de obtener conocimiento en función de las estructuras ontológicas. Para delegar a la ontología la tarea de definir la mejor clase para un individuo determinado se usa ese tipo de software.

En el modelo de clases propuesto se va requerir generar un conjunto de clases que permitan la adecuada inserción de ontologías. Pensemos en la clase Ontología:

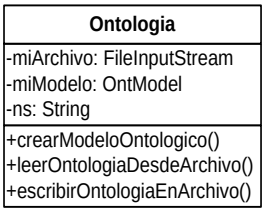


Diagrama 17: Clase Ontología

Esta clase provee la oportunidad de trabajar con un modelo ontológico, es decir facilita la gestión de conceptos especificados en un lenguaje como el OWL.

La clase Ontología requiere una ontología sobre la cual trabajar especificada por el atributo miArchivo, que apunta a un archivo .owl (también se podría optar por consumir una ontología disponible desde internet), que contiene a la ontología. Los archivos .owl, pueden ser escritos directamente en un editor de texto o por medio del uso de software tal como el Protégé [23], detalles sobre esta herramienta se encuentran en el Anexo B “Herramientas para el manejo de ontologías”.

El modelo ontológico se instancia en el atributo miModelo de tipo OntModel, un tipo de objeto que extiende al objeto Model del API RDF de JENA [24], framework empleado para la construcción de aplicaciones semánticas con Java y cuyos detalles se pueden observar en el Anexo B “Herramientas para el manejo de ontologías”. El OntModel toma la ontología especificada en el archivo y la deja accesible para su gestión en una aplicación de software con Java.

El atributo ns, guarda el NameSpace en el cual está almacenada la ontología. Recordemos que las ontologías nacieron como un mecanismo para la implementación de la web semántica y que una de sus propiedades es la oportunidad de compartirlas. Una ontología creada para la gestión de proyectos, podría ser usada por miles de personas y aplicaciones para sus propios fines. El NameSpace de una ontología, permitirá conocer el lugar donde reside el conocimiento y acceder a él.

Cuando se cree un ProductoComponente, se deberá solicitar a la clase Ontología que se proceda a crear un nuevo individuo de la clase ontológica MiDocumento (la clase que engloba a todos los posibles tipos de ProductoComponente), se recomienda usar como nombre el mismo nombre del ProductoComponente para nombrar a la individuo ontológico para en lo posterior identificarlo

como individuo de un ProductoComponente específico (otra opción sería incluir como un atributo de la clase ProductoComponente el nombre de su instancia ontológica). Con la instancia lista para trabajar, se solicita a la clase Ontología que provea la lista de todas las propiedades que posee la clase MiDocumento para establecer los valores de aquellas que sean de interés para el ProductoComponente presentado.

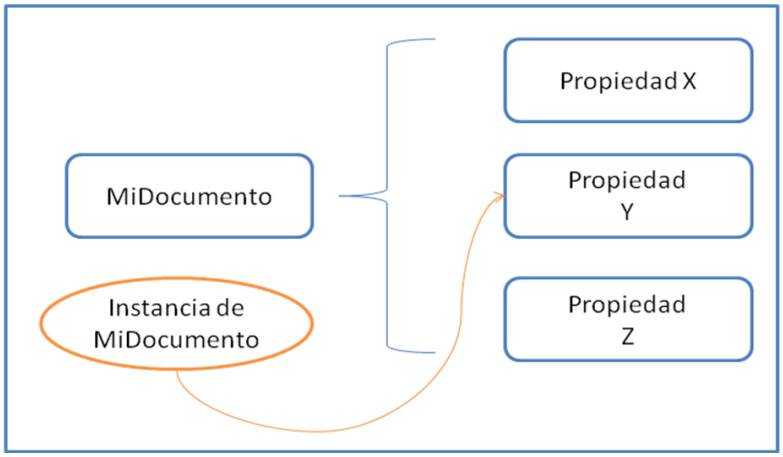


Ilustración 22: Propiedades de la Clase miDocumento

En la ilustración se puede ver como la clase ontológica miDocumento, posee las propiedades X, Y y Z, se muestra además una instancia de esta clase, a la cual se le ha consignado únicamente la propiedad Y. Esta situación va a ser muy común y quien va a decidir qué propiedad tiene valor para un ProductoComponente será el usuario de la aplicación.

Recordemos que una propiedad se puede pensar como una relación que existe entre dos instancias de alguna o algunas clases ontológicas, por lo que si bien, se selecciona la propiedad que afecta a la instancia de la clase ontológica miDocumento que se está creando, se deberá en adición indicar con qué individuo específico se relaciona. El Documento A, presentado en un ejemplo anterior, tenía asignado dos propiedades (además del hecho de ser un documento): TieneAutor y TieneReferenciaAElemento. En el primer caso, la propiedad se enlazó a la instancia de la clase Analista llamada Jorge y en el segundo con el Requerimiento Funcional: mantener el registro actualizado de pasajeros. Dicho de otra manera, va a ser requerido especificar el individuo con el que se relaciona la instancia de miDocumento a través de la propiedad que se haya seleccionado.

Con la lista de propiedades de la clase ontológica miDocumento que provee la clase Ontología, es posible indagar en las clases que conforman el rango de la propiedad y una a una recuperar los individuos existentes, de esta manera se podrá especificar sin ambigüedad la relación existente entre dos individuos de la ontología.

Creada la instancia de la clase ontológica miDocumento, seleccionadas las propiedades que le convienen y definidos los individuos con los que se relaciona a través de cada una de las propiedades, el individuo es insertado en la ontología.

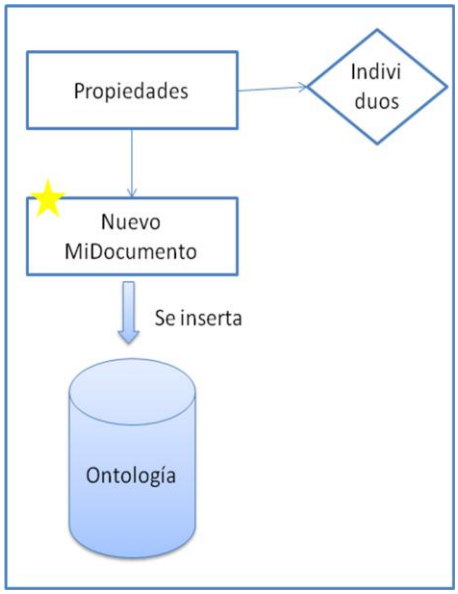


Ilustración 23: Inserción de un individuo de miDocumento

Presentar un ProductoComponente a la ontología y esperar que esta le conceda una clase ontológica acorde a sus propiedades, es solo la primera parte del uso de ontologías para definir un conjunto de información relacionada a un tema. En realidad, lo único que se ha hecho hasta este punto es clasificar a los ProductoComponente, en relación a lo descrito para las folcksonomías estaríamos hasta la etiquetación. Resta entonces encontrar una manera para que esa clasificación sea de utilidad a la hora de seleccionar información de interés.

Para avanzar con la consecución del objetivo, debemos seleccionar uno de los individuos de la clase MiDocumento para hallar información relacionada a él.

En el modelo propuesto, significaría seleccionar un ProductoComponente con el cual trabajar y del cual es de utilidad conocer la información de interés. Debemos tener claro que la información de

interés de un ProductoComponente es otro u otros ProductoComponente que de alguna forma se relacionan con el ProductoComponente actual.

En el ejemplo que se viene analizando podemos ver que el Documento C, es un ProductoComponente que puede ser información de interés cuando se esté trabajando con el Documento A, ya que ambos fueron escritos por una misma persona: Jorge.

La clase Ontología será la encargada de proveer la lista de individuos de la clase ontológica MiDocumento. De esa lista se podrá seleccionar el individuo en cuestión.

Es importante definir qué se va considerar como información de interés o relacionada al individuo seleccionado. Para este trabajo se optó por:

"La información de interés para un individuo en particular, se considera a la lista de otros individuos de la clase ontológica miDocumento, que compartan por lo menos una relación a través de una misma propiedad con los mismos individuos con los cuales se relaciona el individuo que está siendo empleado."

Esta es una suposición que se toma como válida para probar la utilidad de ontologías para obtener información relacionada a un elemento, sin embargo puede ampliarse, refinarse y mejorarse para trabajos futuros.

De este modo, se procede a buscar de entre los individuos de la clase ontológica miDocumento, todos aquellos que compartan propiedades hacia los mismos individuos que el individuo seleccionado.

Para conseguir esa lista, se recurre a la creación de una clase ontológica que posea una propiedad a la vez, la propiedad será una de las propiedades que posee el individuo seleccionado. Estas clases no serán introducidas en la ontología y solo servirán mientras dure el proceso de selección de información de interés, luego se procederá a eliminarlas.

El Documento A ha sido seleccionado por lo que urge obtener la lista de información relacionada. El sistema creará una clase ontológica por cada propiedad del individuo y además al crear la propiedad para la nueva clase ontológica se indicarán los individuos con los cuales se relaciona a través de la propiedad. Esto resultará en dos clases ontológicas:

- Clase A, con propiedad TieneAutor relacionada al individuo Jorge

- Clase B, con propiedad TieneReferenciaAElemento relacionada con Mantener el registro actualizado de pasajeros

Las nuevas clases ontológicas serán clases hijas de la clase miDocumento.

Con las nuevas clases listas, se procede a solicitar a la clase Ontología que devuelva la lista de individuos de cada una de ellas. Es evidente que no poseen instancias creadas, pero gracias a la inferencia, la ontología podrá categorizar a los individuos existentes de otras clases ontológicas como individuos de la clase indicada. Un individuo de la Clase A es el Documento C, pues su autor es Jorge.

De este modo, es factible obtener una lista de individuos relacionados al individuo actual y por tanto proveer de información de interés para quien esté trabajando con un ProductoComponente determinado.

Aplicando el modelo de clases como extensión de sistemas

1.1. Introducción

Los diagramas 16 y 17, presentan un conjunto de clases que podrían emplearse para la construcción de un software orientado a brindar soporte a la Administración de Proyectos, por supuesto que esta construcción implica la creación de una aplicación desde cero, sin embargo, es factible emplear ciertos fragmentos del modelo definido en capítulos anteriores para incorporar la funcionalidad descrita, en un software orientado a la gestión de proyectos que ya exista. Este capítulo realiza un análisis respecto a aplicar el modelo de clases como extensión de sistemas.

1.2. Fragmentos esenciales

Cuando se posee un software de administración de proyectos ya existente, va a ser requerido realizarle algunos ajustes para conseguir incluir las funcionalidades descritas en este proyecto de tesis, desde ese punto de vista lo primero que se va a hacer es definir las clases del modelo propuesto que deberían emplearse en otros sistemas y cómo hacer que funcionen.

De los diagramas 16 y 17 se deberá considerar al juego de clases Observable, Observador y Notificador para integrar al software destino de tal forma de que pueda conocer los cambios contextuales de alguno de los componentes. Ahora bien, para que sea posible responder a estos cambios contextuales con una reacción acorde al nuevo contexto, se requiere trabajar con la clase EstadoContextual y con la jerarquía de clases AccionATomar, que tendrá las clases hijas requeridas según las necesidades establecidas, por ejemplo, si se va a reaccionar a través de correos electrónicos será de interés la clase Mail.

Por otro lado, está la integración de las clases que dan soporte para la clasificación de contenidos que son Etiqueta, para el tema de las folcksonomías, y la clase Ontología.

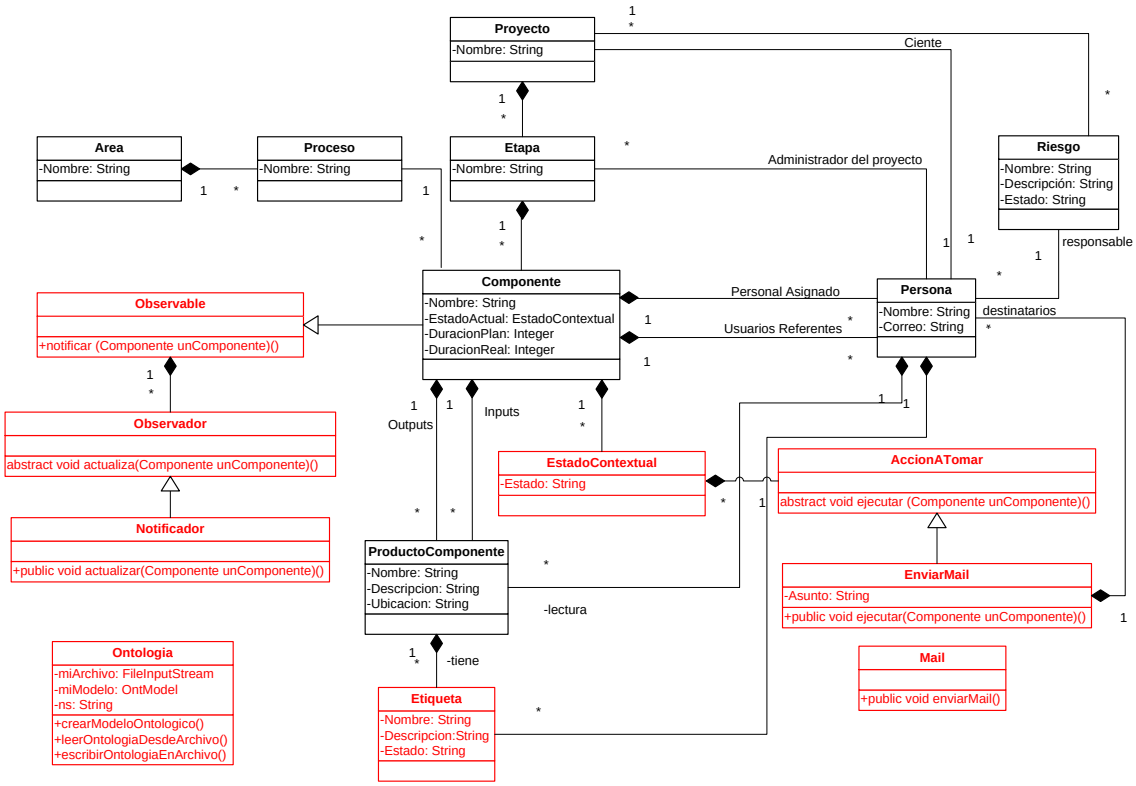


Diagrama 18: Clases Esenciales para extender software

1.2.1. Cambios Contextuales

Continuando, analicemos el software destino y definamos cuál será la clase que contendrá los objetos de los cuales nos interesa saber cuando su contexto ha cambiado. En el modelo de clases propuesto, esa clase de objetos se representa por medio de la clase Componente, una clase de tipo Aware-Object, es decir que conoce qué hacer ante un cambio en su contexto, sin embargo es importante notar que no todos los sistemas de administración de proyectos van a tener definida una clase Componente, pero sin lugar a duda poseerán una clase que les permita manejar las actividades de un proyecto. Como ejemplo genérico podemos hablar de una clase llamada Item.

La clase Item de un software destino, estará constituida por atributos y métodos acorde a las necesidades del software y no de aquellos atributos y métodos requeridos para que el modelo propuesto funcione. La intención, no es ingresar al código de esta clase y modificarlo, sino más bien considerar algún tipo de estrategia que no requiera dichas modificaciones.

Una alternativa podría ser considerar a la referida clase Item como padre de una jerarquía que permita construir un aware-model, cuya clase hija será una denominada Componente, misma que conocemos en nuestro modelo como una clase sensible al contexto.

Componente, heredará toda la funcionalidad de la clase Item, pero adicionalmente permitirá definir los atributos y métodos requeridos para que la clase Item pueda de alguna manera reaccionar ante cambios contextuales.

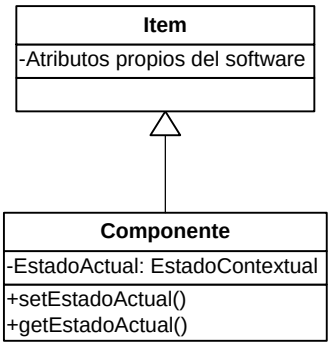


Diagrama 19: Aware-Model Componente

De esa forma Componente extiende la funcionalidad de Item y será la clase a emplear en el software cuando se requiera trabajar con la sensibilidad al contexto, esto sin necesidad de cambiar el código de la clase Item.

Item, mantendrá todas las relaciones con otras clases del modelo del software destino y la clase Componente traerá consigo las relaciones requeridas para que se pueda aplicar la sensibilidad al contexto de la manera en que se puede apreciar en el siguiente diagrama.

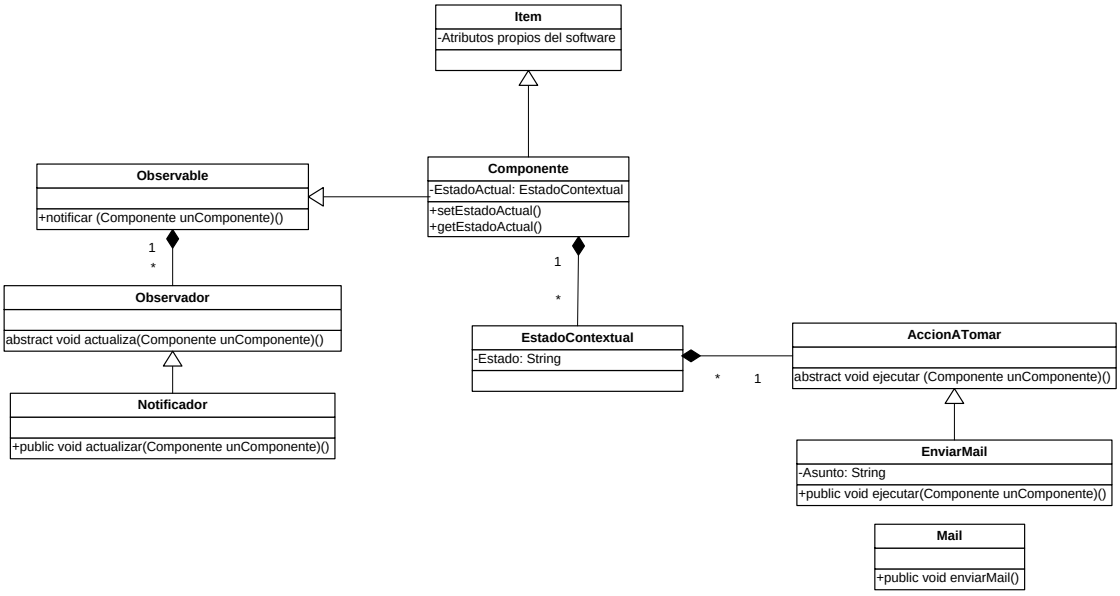


Diagrama 20: Herencia Multiple

El diagrama 20 muestra una solución a la integración de sensibilidad de contexto a un software de administración de proyectos existente. De cualquier manera es evidente la existencia de una herencia múltiple en la clase Componente (hereda de Item y de Observable), que puede traer consecuencias negativas que este tipo de práctica produce. En ese sentido, se ha pensado que se puede construir al Aware-Model Componente de una manera diferente, sin aplicar herencia del objeto Item, más bien envolverlo. De esa manera la clase Componente, contendrá una instancia de Item y adicionalmente contará con los métodos requeridos para la sensibilidad al contexto y de las relaciones del modelo propuesto. Para acceder desde Componente al Item, bastará con referirse a la instancia y acceder a los métodos requeridos.

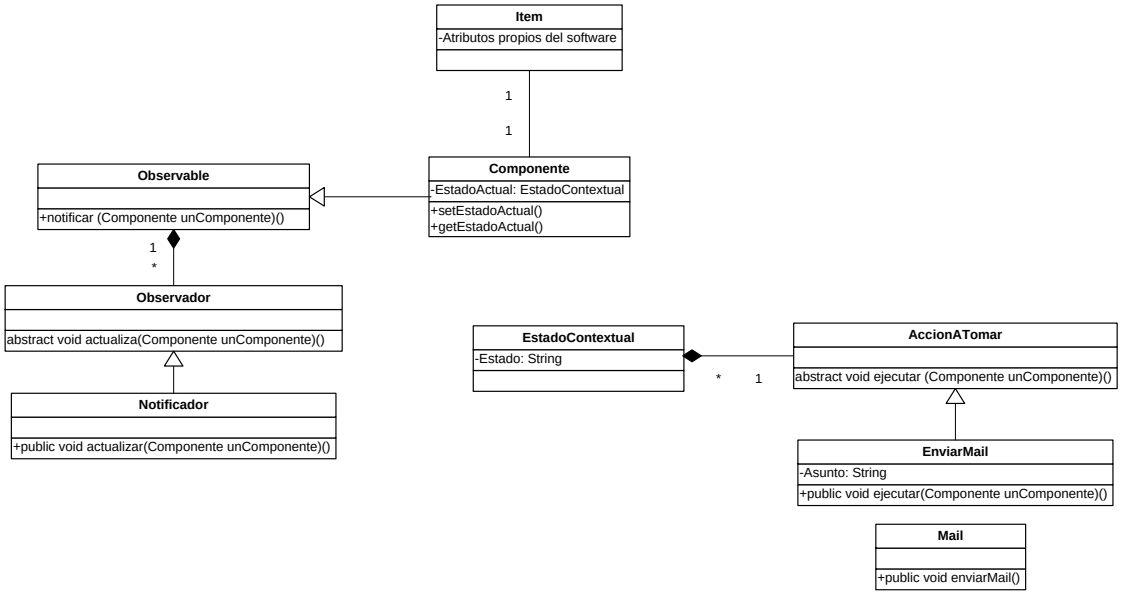


Diagrama 21: Aware- Model Componente

Para mantener el principio de encapsulamiento, la clase Componente debería crear métodos que sirvan para llamar al correspondiente método de la clase Item. Supongamos que la clase Item tiene implementado un método llamado calcularTiempo(), para acceder al mismo desde la clase Componente, se deberá crear un método llamado calcularTiempoItem(), que en realidad ejecute el método calcularTiempo() de la clase Item, a través de la instancia de Item que Componente conoce. Los parámetros de los métodos deberán usarse iguales a los definidos en la clase Item.

El software destino, no deberá cambiar sus instanciaciones del objeto Item, por el de Componente, ya que el objeto Item no ha cambiado en lo absoluto. Lo que hay que tomar en cuenta es que cuando se desee tener una funcionalidad sensible al contexto se deberá usar la instancia de Componente que conoce al Item en cuestión.

1.2.2. Etiquetado Folcksonómico

Cuando se describía el modelo propuesto, se habló de la posibilidad de brindar información de interés a los usuarios, en relación al ítem actual y conforme un determinado contexto, y se vio que una manera de conseguirlo era por medio de folcksonomías que empleaban la clase Etiqueta.

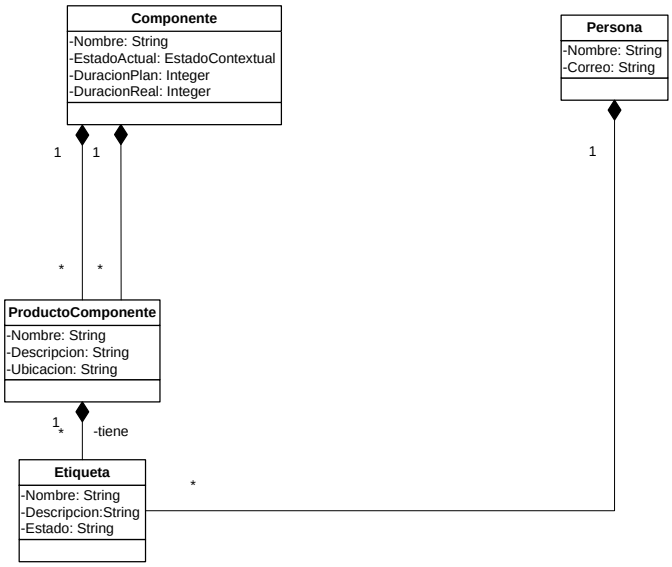


Diagrama 22: Diseño para Folcksonomías

Los componentes conocían sus inputs y sus correspondientes outputs a los que se les representó con la clase ProductoComponente. Cada instancia de esta clase podía poseer una colección de etiquetas folcksonómicas colocadas por alguna persona usuaria del software.

Para adaptar esta funcionalidad a un software destino, se deberá en primer lugar verificar la existencia de alguna clase que se relacione al Item por medio de las relaciones inputs y outpus o sus equivalentes en el modelo del software destino.

Este diseño puede no estar presente en el software destino por lo que se podría aplicar directamente a la clase Item, esta decisión significará que las etiquetas se consignarán a nivel de Item y no de sus productos.

Luego de definida la clase a la cual se aplicarán las etiquetas, el siguiente paso consiste en crear una clase que conozca a qué objeto se aplica la etiqueta y el conjunto de etiquetas aplicadas, y claro, añadir la clase Etiqueta.

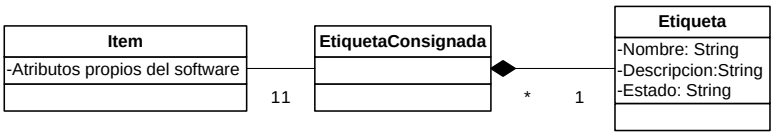


Diagrama 23: Extendiendo Folcksonomías

La clase EtiquetaConsignada, facilita el hecho de no modificar las clases del modelo del software destino, sin decrecer en funcionalidad.

Para conocer el propietario de una etiqueta será necesario verificar la existencia de una clase de Usuario o Persona o una similar dentro del modelo del software destino, con la cual enlazar a la clase Etiqueta. Dicho enlace se realizará por medio de la creación de una nueva clase llamada MisEtiquetas que conocerá al propietario y al conjunto de etiquetas que le pertenecen.

1.2.3. Ontologías

Las ontologías se aplicaron también para obtener información de interés, en el caso de usarlas como extensión su aplicación será la misma que cuando se construye un software desde cero. Habrá que usar la clase Ontología, crear una ontología, capaz de guardar el concepto de los términos empleados en el software destino y aplicar los procedimientos definidos para hacer que la ontología infiera cuáles serían informaciones de interés. En definitiva esta extensión no requiere de trabajo adicional.

El Prototipo

1.1. Introducción

Durante los capítulos de este trabajo de tesis se ha hablado, analizado y definido un modelo de clases capaz de dar solución a algunos inconvenientes detectados en las herramientas diseñadas para la Administración de Proyectos. El capítulo actual resume los detalles de implementación de ese modelo a manera de un prototipo orientado a probar su utilidad.

El modelo de clases definido se ha implementado parcialmente en un prototipo desarrollado en el lenguaje de programación JAVA, haciendo uso del plugin Jena-2.6.4, para el manejo de ontologías y de Pellet-2.2.2 para la inferencia de una ontología, además de varios paquetes de clases desarrollados para JAVA.

La intención del prototipo es exclusivamente probar la funcionalidad del modelo de clases, por lo que, los detalles relacionados a su interfaz no son tema de interés resultando de esta manera el desarrollo de una aplicación para consola.

Una aplicación para Administración de Proyectos requiere la creación de una multitud de instancias a través de los respectivos manejadores de ABM's, lo que resulta inaplicable para pruebas, razón por la que se optó por la creación de un conjunto de instancias de ejemplo a las que inicializa y con las que se trabaja en las diversas pruebas, vea el Anexo C "Instancias" para mayores detalles.

1.2. Menú

El menú definido para el prototipo posee las siguientes opciones:

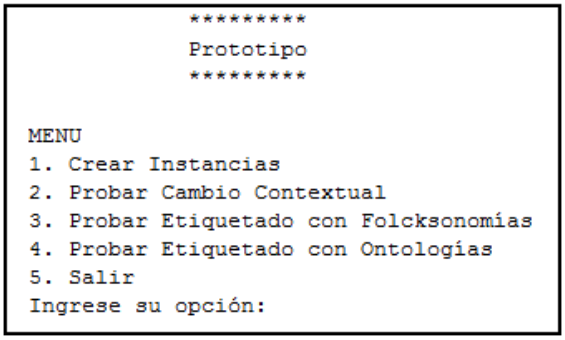


Imagen 5: Menú principal del prototipo

La primera opción del menú, se refiere a la ejecución de un método que crea el conjunto de instancias de ejemplo para el prototipo. Esta opción es de ejecución obligatoria previa la selección de cualquiera de las otras opciones del menú a excepción de la opción 5. Salir.

La opción: Probar Cambio Contextual, pretende verificar que es factible ejecutar una serie de acciones ante cambios contextuales de un ProductoComponente.

La tercera opción, se enfoca al proceso de etiquetado por medio de folcksonomías y luego el proceso de obtener información relacionada haciendo uso de estas etiquetas.

La opción número cuatro: Probar Etiquetado con Ontologías, hace uso de una ontología de ejemplo diseñada en owl y almacenada en un archivo, sobre la cual se definen conceptos a ser aplicados a los productos componentes de un proyecto, luego haciendo uso de esas definiciones se determina un conjunto de información de interés para un tema particular.

La opción salir, finaliza la aplicación.

Las secciones siguientes detallan la implementación:

1.3. Opción: Probar Cambio Contextual

Esta opción del menú tiene la intención de realizar una prueba sobre la Aplicación de Sensibilidad al Contexto en sistemas de Administración de Proyectos.

La prueba consiste en demostrar que a través del modelo de clases del capítulo “Aplicando Sensibilidad al Contexto a sistemas para la Administración de Proyectos”, es posible ejecutar una

serie de acciones al momento en que se da un cambio en el estado contextual de uno de los componentes del proyecto.

Consideremos a la instancia de la Clase Componente llamada “Creación de Realizaciones de Casos de Uso”, a la cual se le ha asignado un único Estado Contextual de interés llamado “Creación” (recordemos que un componente puede tener n o 0 estados contextuales de interés). Cuando el componente alcanza dicho estado contextual, se ha configurado la ejecución de dos acciones: presentación de información relacionada y envío de correo a una lista de interesados.

Cuando se selecciona esta opción del menú, el sistema permite seleccionar: la etapa a la cual pertenece el componente, el componente y solicita se indique el nuevo estado contextual al que llegaría el componente (este trabajo en realidad sería de los sensores, pero para el caso se especifica de forma manual). Teniendo seleccionado el componente se establece el nuevo estado contextual:

unComponente.setEstadoActual(elEstadoActual);

Esta sentencia ejecuta el método de la clase Componente setEstadoActual:

```
public void setEstadoActual(EstadoContextual unEstado)
 {
                EstadoActual = unEstado;
                this.notificar(this);
}
```

La sentencia this.notificar(this) ejecuta el método notificar de la clase Observable, clase de la cual es hija Componente. Hay que considerar que la clase Observable es abstracta.

```
public void notificar(Componente unComponente)
{
        Iterator <Observador> it = this.observadores.iterator();
        while (it.hasNext())
                it.next().actualizar(unComponente);
}
```

El método itera sobre la lista de observadores que se han suscrito para obtener novedades sobre el componente en cuestión. Por cada observador suscrito, se ejecuta el método actualizar de la clase Observador, o mejor dicho, de su observador concreto, en el modelo: la clase Notificador.

```
public  void actualizar(Componente unComponente)
{
Iterator <AccionATomar> at = unComponente.getEstadoActual().getAcciones().iterator();
        AccionATomar unaAccion;
        while (at.hasNext())
        {
                unaAccion = at.next();
                unaAccion.ejecutar(unComponente);
        }
}
```

El método obtiene la lista de acciones a tomar e itera sobre ellas, haciendo que cada una de las acciones ejecute su método ejecutar. En el ejemplo, las acciones definidas son proveer información y enviar email, de esta manera la primera acción será instancia de la clase ProveerInformación y la segunda de EnviarMail.

Finalmente, el método ejecutar de cada una de las clases realizará el trabajo definido como respuesta a un cambio contextual en el componente.

Al correr el prototipo, podemos obtener esta información:

```
Desea seleccionar el componente Creación de Realizaciones de Casos de Uso s/n?
s
Ingrese el estado contextual:
Creacion
Accion a Tomar: Notificar cambio a estado: Creacion
Presentando informacion relacionada al componenete Creación de Realizaciones de Casos de Uso
Se ha enviado un mail a Antonio sobre: Se han creado nuevos documentos de Realizacion de Casos de Uso
Se ha enviado un mail a Jorge sobre: Se han creado nuevos documentos de Realizacion de Casos de Uso
```

Imagen 6: Selección de un estado contextual para un componente

Con ello, se demuestra que el modelo es capaz de ejecutar acciones pre configuradas para dar respuesta a un cambio contextual de un componente de un proyecto.

El detalle de la presentación de la información relacionada así como el envío de correo electrónico no es de interés para esta prueba.

Por otro lado, si bien se optó por la utilización de acciones concretas como ProveerInformación y EnviarMail, la forma de trabajo asegura el buen desempeño para otros tipos de acciones que pudieran llegar a definirse, como agendar, publicar entre otras, ya que el método ejecutar (definido en cada una de ellas), es el encargado final de realizar las acciones respectivas.

1.4. Opción: Probar Etiquetado con Folcksonomías

Esta opción del menú permite verificar el funcionamiento del etiquetado en los elementos ProductoComponente de un Proyecto, descrito en el capítulo "Métodos para obtener información de interés" en la sección "Usando Folcksonomías".

Para esta prueba se ha considerado una instancia de la clase Persona, que se supone está autenticada en el sistema, y por lo tanto será el responsable de las acciones ejecutadas en torno a las folcksonomías. Para ello se definió en la clase main del prototipo la siguiente sentencia:

protected static Persona usuario;

Durante el proceso de instanciación, se seleccionó a la persona "José" como el usuario autenticado.

Cuando se ingresa en esta opción del menú el software presenta las siguientes alternativas:

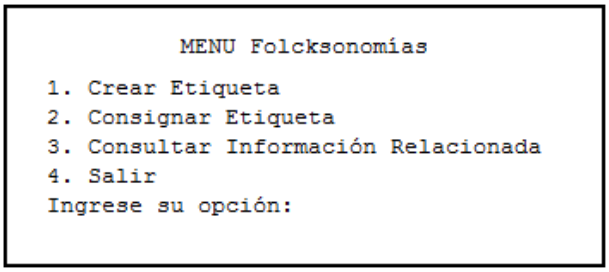


Imagen 7: Opciones del submenú de pruebas de folcksonomías

La primera opción, permite la creación de una nueva etiqueta folcksonómica en el sistema, cuyo propietario es el usuario actual del software (en el caso del prototipo, la persona: José).

El proceso de creación de una etiqueta solicita las características de la misma, tal como nombre y descripción, además por defecto se configura al estado como 0 que se refiere a activo. La creación de etiquetas se lo hace por medio del método crearEtiqueta() de la clase main.

En este método se invoca a los métodos setNombre(), setDescripcion() y setEstado() de la clase Etiqueta.

unaEtiqueta.setNombre(miScanner.nextLine());

unaEtiqueta.setDescripcion(miScanner.nextLine());

unaEtiqueta.setEstado(0);

Cada vez que se crea una nueva etiqueta es agregada a la colección de etiquetas del usuario autenticado:

Etiquetas = usuario.getEtiquetas();

.

.

.

etiquetas.add(unaEtiqueta);

Una nueva etiqueta en la colección de etiquetas de un usuario, queda disponible para ser usada en el tagging de documentos.

La alternativa Consignar Etiqueta del menú de Folcksonomías, permite al usuario autenticado etiquetar algún documento con una de las etiquetas de su propiedad. El sistema permitirá seleccionar el documento al cual etiquetar, y una vez elegido procederá a indicar las etiquetas (aquellas de propiedad del usuario), que puede consignar al documento:

```
Desea consignar una etiqueta a este ProductoComponente Lista de usuarios referentes s/n?
n
Desea consignar una etiqueta a este ProductoComponente Documento justificacion del proyecto s/n?
n
Desea consignar una etiqueta a este ProductoComponente Lista de requerimientos s/n?
s
Desea consignar la etiqueta Ayuda para principiantes s/n?
n
Desea consignar la etiqueta prueba s/n?
s
```

Imagen 8: Selección de una etiqueta folcksonómica para un ProductoComponente

Al indicar al sistema la etiqueta seleccionada para el documento se ejecuta la sentencia:

unasEtiquetas.add(unaEtiqueta);

unasEtiquetas es la colección de etiquetas que posee el ProductoComponente, es decir el conjunto de etiquetas que se le haya dado sin importar cuál fue su autor.

La opción número tres: Consultar Información Relacionada, facilita la ubicación de un documento del cual se presentará la información relacionada, en función de sus etiquetas.

```
Desea seleccionar el ProductoComponente Lista de usuarios referentes s/n?
n
Desea seleccionar el ProductoComponente Documento justificacion del proyecto s/n?
n
Desea seleccionar el ProductoComponente Lista de requerimientos s/n?
n
Desea seleccionar el ProductoComponente Glosario s/n?
n
Desea seleccionar el ProductoComponente CU 1. Manejar ABM de Destinos s/n?
s
```

Imagen 9: Ubicación de un ProductoComponente

Los métodos consultarInformacionConFolcksonomias() y buscarProductosComponentes() de la clase main, son los encargados de interactuar con las clases ProductoComponente y Etiqueta para obtener los documentos que mantengan relación con algún documento en particular.

Al seleccionar el ProductoComponente del cual se desea indagar la información relacionada se obtiene la lista de etiquetas que le han sido consignadas:

```
misEtiquetas = unProductoComponente.getMisEtiquetas();
```

Luego, por cada etiqueta se procede a buscar los Productos Componentes que también han sido etiquetados con la misma tag.

```
misEtiquetas = unProductoComponente.getMisEtiquetas();
iet = misEtiquetas.iterator();
while(iet.hasNext())
{
        laEtiqueta = (Etiqueta)iet.next();
        if (laEtiqueta == unaEtiqueta)
                informacionRelacionada.add(unProductoComponente);
}
```

La colección InformaciónRelacionada guarda todas las instancias de ProductoComponente que poseen al menos 1 etiqueta igual a nuestro documento de interés.

Finalmente el sistema presentará, ordenados por etiqueta, los documentos relacionados:

```
Informacion Relacionada a: CU 1. Manejar ABM de Destinos

Por la etiqueta Ayuda para principiantes:
    Lista de usuarios referentes
    Lista de requerimientos
    Glosario
    CU 6. Vender Boletos
    RCU 5. Manejar ABM de azafatas


Por la etiqueta Mis Casos de Uso:
    CU 2. Manejar ABM de Orígenes
    CU 3. Manejar ABM de Pilotos
    CU 4. Manejar ABM de Vuelos
    CU 5. Manejar ABM de Azafatas
    CU 6. Vender Boletos
```

Imagen 10: Información Relacionada obtenida con el uso de Folcksonomías

Así, se puede decir que a través del uso del modelo propuesto en este trabajo de tesis, es posible acceder a sugerencias sobre información relacionada a un elemento particular con el uso de un etiquetado folcksonómico.

1.5. Opción: Probar Etiquetado con Ontologías

Esta opción del menú permite verificar el funcionamiento de la asignación de una etiqueta ontológica en los elementos ProductoComponente de un Proyecto, descrito en el capítulo “Métodos para obtener información de interés” en la sección “Usando Ontologías”.

Para esta prueba es requerido el concurso de una ontología que describa el dominio de la Administración de Proyectos, obviamente es una versión escueta, suficiente para realizar las pruebas requeridas.

La ontología se creó utilizando el software Protégé, consiguiendo el siguiente árbol de clases:

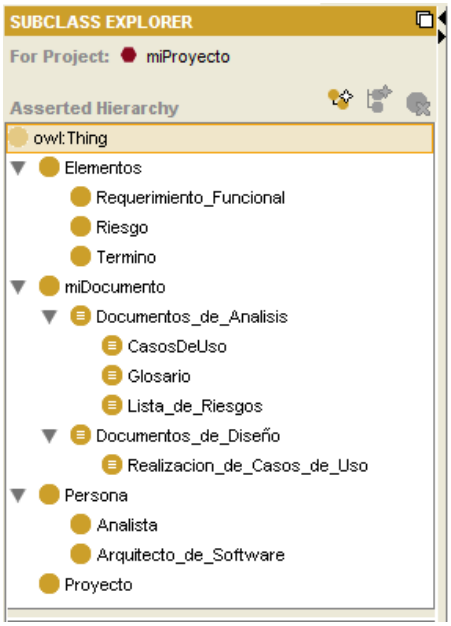


Imagen 11: Arbol de clases ontológicas

El proyecto de Protégé miProyecto.pprj, guarda los detalles de la ontología de prueba. Desde Protégé, se consiguió el archivo miProyecto.owl, que especifica la ontología en lenguaje OWL DL, lenguaje con el suficiente nivel de detalle para emplear un razonador sobre la ontología. Para ver la ontología de prueba completa vea el Anexo A "Ontología de Prueba".

Cuando se ha seleccionado esta opción del menú el software presenta las siguientes opciones:

```
          MENU Ontologías
1. Presentar ProductoComponente a Ontología
2. Consultar Información Relacionada
3. Salir
Ingrese su opción:
```

Imagen 12: Opciones para la prueba de Ontologías

La primera opción permite presentar un ProductoComponente a la ontología, o dicho de otra manera permite crear un individuo de la clase miDocumento en la ontología de prueba.

Lo primero que se deberá hacer es crear el modelo ontológico, para lo cual el método crearOntologia(String ns) de la clase main es el encargado de interactuar con la clase Ontología para obtener una instancia de un OntModel creado así:

this.miModelo = ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);

El valor PelletReasonerFactory.THE_SPEC es usado para indicar que el modelo ontológico que se está creando soporta al razonador Pellet [25].

Luego se procede a leer la ontología desde el archivo:

this.miModelo.read(this.miArchivo, "");

Despúes se obtiene la clase ontológica miDocumento, recordemos que es la clase padre de cualquier tipo de ProductoComponente en la ontología y que es usada para crear sin caer en detalles individuos de las clases ontológicas en la ontología:

claseDocumento = miOntologia.getClase("http://www.miProyecto.owl#miDocumento");

El software procede a seleccionar el ProductoComponente que se presentará a la ontología, para ello se recorrerán las etapas y componentes en busca del ProductoComponente requerido. El método seleccionarProductoComponente() de la clase main es el encargado de devolver miProductoComponente:

miProductoComponente=seleccionarProductoComponente(unProyecto);

Este es el momento de crear un individuo de la clase miDocumento para el ProductoComponente seleccionado, para lo que se solicita a la ontología la ejecución de su método crearInstancia():

```
public Individual crearInstancia( OntClass clase, String nombre)
        {
                //creo la instancia nueva de la clase enviada como parametro
                return clase.createIndividual(this.ns+nombre);
        }
```

Llamándolo así:

nuevoIndividuo = miOntologia.crearInstancia(claseDocumento, miProductoComponente.getNombre());

Con el nuevo individuo creado, se procede a obtener todas las propiedades de la clase miDocumento y luego se procede a recorrer cada propiedad para conocer cuál de ellas es de interés para el ProductoComponente presentado. El prototipo presenta la siguiente información:

```
Desea seleccionar el ProductoComponente CU 1. Manejar ABM de Destinos s/n?
s
Desea agregar la propiedad: TieneReferenciaAAnalisis? s/n
n
Desea agregar la propiedad: PerteneceAProyecto? s/n
n
Desea agregar la propiedad: TieneAutor? s/n
s
```

Imagen 13: Selección de la propiedad a definir par aun ProductoComponente

Cuando se ha seleccionado una propiedad, el software procede a buscar los individuos de todas las clases que conforman el rango de clases definidas para la propiedad. Recordemos que el rango de una propiedad son las clases con cuyos individuos se puede relacionar un individuo de una clase que posee la propiedad.

```
Desea agregar la propiedad: TieneAutor? s/n
s
Desea agregar a la propiedad individuos de la clase: Arquitecto_de_Software s/n:
n
Desea agregar a la propiedad individuos de la clase: Analista s/n:
s
Desea agregar a: Emilio s/n? n
Desea agregar a: Mario s/n? s
Desea agregar la propiedad: TieneReferenciaAElemento? s/n
```

Imagen 14: Selección de individuos para relacionar

Al seleccionar la propiedad de interés, el software procede a buscar individuos clasificados por cada una de las clases del rango de la propiedad, es así que en la imagen se pueden apreciar las clases Arquitecto_de_Software y Analista. Seleccionada la clase del rango se muestran sus individuos. Al decidirse por un individuo, entonces se estaría creando una propiedad para el nuevo individuo de la clase miDocumento.

Para soportar la creación de propiedades para un individuo se ha creado una clase utilitaria llamada Propiedad con los siguientes atributos:

protected Individual unIndividuo;

protected Individual elIndividuo;

protected OntProperty unaPropiedad;

Un individuo podrá tener una colección de propiedades que servirán para crear al individuo en la ontología. Se usó la colección nuevas en la clase main:

protected static Collection<Propiedad> nuevas;

Cuando la colección contenga todas las propiedades seleccionadas se procede a recorrerlas una a una y a definirlas en la ontología a través del método atarInstancias() de la clase Ontología:

```
public  void atarInstancias(Propiedad nuevaPropiedad )
        {

        nuevaPropiedad.getElIndividuo().addProperty(nuevaPropiedad.getUnaPropiedad(),nuevaPropied
ad.getUnIndividuo());
        }
```

Terminado ese proceso, se escribe el nuevo individuo en la ontología:

```
miOntologia.escribirOntologiaEnArchivo();
```

De esta manera es posible presentar un ProductoComponente a la ontología y posterior a este instante el individuo queda disponible para cualquier trabajo relacionado a la ontología. Cabe recalcar que la escritura de la ontología se lo realiza en un archivo de texto plano, el cual será leído cada vez que se inicie la aplicación. Este método (dependiendo del tamaño de la ontología), podría resultar lento, por lo que también hay opciones de persistir la ontología en una base de datos.

Cuando se seleccione la segunda opción del Menú de Ontologías: Consultar Información Relacionada, el programa procede a interrogar a la ontología sobre instancias de la clase miDocumento que sean similares a un elemento determinado.¨

El primer paso que se realiza es seleccionar el individuo del cual se va a obtener información de interés, para ello se recorren todas las instancias de la clase ontológica miDocumento ya que se asume que se tienen a todos los ProductosComponentes presentados a la ontología. El método seleccionarIndividuo() de la clase main, es la encargada de devolver al individuo seleccionado:

```
individuoSeleccionado=seleccionarIndividuo(miOntologia,claseDocumento);
```

```
Desea seleccionar un documento de tipo Documentos_de_Diseño s/n:
n
Desea seleccionar un documento de tipo Documentos_de_Analisis s/n:
s
Desea seleccionar a CU2 ABM Origenes s/n? n
Desea seleccionar a G1 Glosario del proyecto s/n? n
Desea seleccionar a CU4 ABM de vuelos s/n? n
Desea seleccionar a CU6 Vender Boletos s/n? s
```

Imagen 15: Selección del individuo ontológico

Con ese individuo se procede a ejecutar el método presentarInformacionRelacionada() de la clase main. En este método se generan clases ontológicas intermedias o utilitarias para definir clases para cada propiedad del individuo seleccionado:

```
documentoRelacionado = miOntologia.getMiModelo().createClass(miOntologia.getNs()+"dC");
```

Se recorre cada propiedad y para cada una de ellas se definen los miembros de la propiedad, finalmente se crea la propiedad como un HasValueRestriction, que es la manera en que OWL permite indicar que una propiedad apunta a un individuo en particular:

```
s = mp.getPredicate(); //la propiedad
ind = mp.getObject(); //el individuo
hv = miOntologia.getMiModelo().createHasValueRestriction("", s, ind);
documentoRelacionado.addEquivalentClass(hv);
```

Luego, lo que se realiza es consultar a la ontología por individuos que coincidan con la clase recién creada. Cada individuo representa una información relacionada

```
i=miOntologia.getIndividuos(documentoRelacionado);
while (i.hasNext())
{
        pr = (Individual) i.next();
        resultados.add(pr.toString());
        resultados.add("");
}
```

```
Individuo Seleccionado: http://www.miProyecto.owl#CU6
Para la propiedad: http://www.miProyecto.owl#TieneReferenciaAElemento

Informacion Relacionada encontrada
http://www.miProyecto.owl#CU6

Para la propiedad: http://www.miProyecto.owl#TieneReferenciaAElemento

Informacion Relacionada encontrada
http://www.miProyecto.owl#CU6

http://www.miProyecto.owl#CU 2. Manejar ABM de Orígenes
```

Imagen 16: Información relacionada obtenida con el uso de Ontologías

La imagen, muestra como por propiedad, se presenta las clases ontológicas relacionadas, con lo que se prueba la factibilidad del uso de ontologías para presentar información relacionada a un elemento dado.

Con la implementación de una interfaz más detallada, la funcionalidad puede ser mucho más eficiente que la presentada en el prototipo.

2. Conclusiones y Trabajos Futuros

2.1. Conclusiones

La integración de la Sensibilidad al Contexto en sistemas orientados a asistir a la Administración de Proyectos, resultó ser una valiosa herramienta para saldar inconvenientes de este tipo de sistemas, mediante la implementación de un modelo de clases dispuestas para trabajar en entornos de computación ubicua.

Justamente, el hecho de introducir a los sistemas para la Administración de Proyectos en un entorno de computación ubicua implicó poder conocer los detalles de las realidades contextuales de cada uno de sus elementos, y con ello los sistemas se comprometieron a actuar de una forma más coherente con el entorno en el que se desenvuelven, sin embargo queda la expectativa respecto a la calidad y cantidad de información que se pueda llegar a obtener desde los sensores, cuestión que en este trabajo se simuló.

Si bien, se pretende dotar de cierta inteligencia a los sistemas, siempre se ha considerado que la experiencia de un administrador de proyectos es un bien de alta cuantía por lo que se vio conveniente dejar una puerta abierta para que sea este individuo quien defina las acciones a tomar ante un cambio contextual en alguno de los componentes de su proyecto.

La adopción de modelos de clasificación ontológica y folcksonómica, permitieron el manejo de la información que rodea al proyecto, y como se vio, proveyeron un mecanismo a través del cual fue posible seleccionar información actual o del pasado, que pueda resultar de utilidad en una circunstancia determinada.

Finalmente, hay que estar consciente de que el modelo propuesto parte desde cero y no se aplica a un sistema existente al cual modificar en busca de agregarle sensibilidad al contexto, por ello se describieron algunas consideraciones a tomar cuando se pretenda extender a un software existente con el modelo propuesto, ésta extensión procuró modificar en lo menos posible el modelo del software destino, de tal manera de hacer sencilla la extensión de funcionalidad.

2.2. Trabajos Futuros

La Sensibilidad al Contexto es una tecnología que todavía no ha alcanzado su madurez, por lo que podría resultar interesante trabajar en mecanismos que faciliten su implementación tanto en software en producción como en proyectos nuevos.

Otro punto interesante es investigar métodos para la adquisición de la información contextual desde sensores o agentes de software.

En cuanto al uso de ontologías, sería útil generar una ontología para la Administración de Proyectos que sea validada por la comunidad internacional y que se convierta en el lenguaje universal de este dominio.

Se anotó que las folcksonomías carecían de formalidad, para ello podría trabajarse en una interconexión de las folcksonomías con la ontología empleada, de tal manera de que cada etiqueta folcksonómica corresponda a un concepto ontólogico. De esta forma se estaría extrayendo lo mejor de los dos mundos.

Estaría bueno, tomar un software de administración de proyectos y realizar una extensión de funcionalidad a través del modelo propuesto y verificar su utilidad.

Referencias

[1] Chaos Report, Standish Group, 1995. www.standishgroup.com

[2] A guide to the Project Management Body of Knowledge, Project Management Institute PMI, 1996. www.pmi.org

[3] Open WorkBench, producto de Niku. Niké. www.niku.com

[4] PhpProjekt, www.phpprojekt.com

[5] OpenProj, openproj.org

[6] Weiser Mark, The Computer for the Twenty-First Century, Ubicomp, Scientific American, pp. 94-10, 1991.

[7] Schilit B., Adams N. and Want R., Context-aware computing applications, IEE IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94), 1994.

[8] Dey A.K. and Abowd G.D., Towards a better understanding of context and context-awareness, Technical report, Georgia Institute of Technology, 1999.

[9] Brusilovsky P., Adaptive hypermedia, Ten Year Anniversary Issue, Alfred Kobsa, ed. 11 (1/2), 87-110, 2001.

[10] Marmasse N. and Schmandt C., Location-aware, Information delivery with ComMotion. In HUC pp. 157 -171, 2000.

[11] Venkatesh A., Smart Home Concepts: Current Trends, University of California, Irvine, 2003.

[12] Dey A.K. and Abowd G.D., The Context Toolkit: Aiding the Development of Context Aware Applications, Georgia Institute of Technology, 2000.

[13] Hofer T., Schwinger W., Pichler M., Leonhartsberger G., Altmann J., Context-Awareness on Mobile Devices – the Hydrogen Approach, In Proceedings of HICCS, 2003.

[14] Diccionario de la Lengua Española, Vigésima Segunda Edición, www.rae.es

[15] Semantic Web technologies trends and research in ontology-based systems, Davies J. , Studer R. ,and Warren P. , England: John Wiley & Sons, Ltd, 2006.

[16] Gómez Perez M., Asunción C., Fernández López O., Ontological Enineering, Primera Edición, 2004.

[17] RDF Vocabulary Description Language 1.0: RDF Schema,W3C, 2004, www.w3.org/TR/rdf-schema/

[18] OWL Web Ontology Language, W3C, 2004, www.w3.org/TR/owl-features/

[19] Vander Wal T., Folksonomy, 2004, vanderwal.net/folksonomy.html

[20] Delicious, www.deliciuos.com

[21] Flickr, www.flickr.com

[22] Luis Bertol J., Dolado J, Una Ontología para la gestión del conocimiento de proyectos de software, Universidad de Zaragoza y Universidad del País Vasco, 2008.

[23] Protege, http://protege.stanford.edu/

[24] JENA, http://jena.sourceforge.net/index.html

[25] Pellet, http://clarkparsia.com/pellet/

[26] P. Castells, La Web Semántica, Universidad Autónoma de Madrid, 2005.

[27] SWAD-Europe, Semantic Web Advanced Development for Europe, http://www.w3.org/2001/sw/Europe/

[30] Resource Description Framework, http://es.wikipedia.org/wiki/Resource_Description_Framework.

[31] J. Davies, D. Fensel, y F. V. Harmelen, Eds., Towards the Semantic Web: Ontology-Driven Knowledge Management. England: John Wiley & Sons, Ltd, 2003.

[32] M. Horridge, A Practical Guide to Build OWL Ontologies, 2004.

[33] Leighton H., Berlanga A. García J., La Interacción en los Sistemas Hipermedia Adaptativos, Un Efoque Cognitivo.

[34] Fortier A., Gordillo S., Rossi G., Arquitectural Abstractions for UBICOMP Frameworks, LIFIA.

Anexo A Ontología de Prueba

El empleo de las ontologías para proveer información de interés dentro de este trabajo, requirió el uso de un conjunto de conceptos ontológicos para probar la funcionalidad.

Vale recalcar que la ontología creada no siguió ningún método formal para su conformación, sino se definió orientándose a su uso en un conjunto de pruebas.

La ontología se conformó de la siguiente manera:

El archivo owl define todos los conceptos y sus propiedades anexas. A continuación se presenta la ontología escrita en OWL DL con el soporte de Protégé:

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
  xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:swrl="http://www.w3.org/2003/11/swrl#"
  xmlns="http://www.miProyecto.owl#"
  xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xml:base="file:/D:/miProyecto1.owl">
```

```
<owl:Ontology rdf:about="http://www.miProyecto.owl"/>
<owl:Class rdf:about="http://www.miProyecto.owl#Glosario">
 <rdfs:subClassOf>
  <owl:Class rdf:about="http://www.miProyecto.owl#Documentos_de_Analisis"/>
 </rdfs:subClassOf>
 <owl:equivalentClass>
  <owl:Restriction>
   <owl:onProperty>
    <owl:ObjectProperty rdf:about="http://www.miProyecto.owl#TieneReferenciaAElemento"/>
   </owl:onProperty>
   <owl:someValuesFrom>
    <owl:Class rdf:about="http://www.miProyecto.owl#Termino"/>
   </owl:someValuesFrom>
  </owl:Restriction>
 </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:about="http://www.miProyecto.owl#Documentos_de_Analisis">
 <owl:equivalentClass>
  <owl:Class>
   <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="http://www.miProyecto.owl#TieneReferenciaAElemento"/>
     </owl:onProperty>
     <owl:someValuesFrom>
      <owl:Class rdf:about="http://www.miProyecto.owl#Elementos"/>
     </owl:someValuesFrom>
    </owl:Restriction>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="http://www.miProyecto.owl#TieneAutor"/>
     </owl:onProperty>
     <owl:someValuesFrom>
      <owl:Class rdf:about="http://www.miProyecto.owl#Analista"/>
     </owl:someValuesFrom>
```

```
      </owl:Restriction>
     </owl:intersectionOf>
    </owl:Class>
   </owl:equivalentClass>
   <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.miProyecto.owl#miDocumento"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.miProyecto.owl#Documentos_de_DiseÃ±o">
   <rdfs:subClassOf rdf:resource="http://www.miProyecto.owl#miDocumento"/>
   <owl:equivalentClass>
    <owl:Class>
     <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="http://www.miProyecto.owl#TieneAutor"/>
       </owl:onProperty>
       <owl:someValuesFrom>
        <owl:Class rdf:about="http://www.miProyecto.owl#Arquitecto_de_Software"/>
       </owl:someValuesFrom>
      </owl:Restriction>
      <owl:Restriction>
       <owl:someValuesFrom rdf:resource="http://www.miProyecto.owl#Documentos_de_Analisis"/>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="http://www.miProyecto.owl#TieneReferenciaAAnalisis"/>
       </owl:onProperty>
      </owl:Restriction>
     </owl:intersectionOf>
    </owl:Class>
   </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://www.miProyecto.owl#Realizacion_de_Casos_de_Uso">
   <owl:equivalentClass>
    <owl:Restriction>
     <owl:someValuesFrom>
```

```
      <owl:Class rdf:about="http://www.miProyecto.owl#CasosDeUso"/>
    </owl:someValuesFrom>
    <owl:onProperty>
      <owl:ObjectProperty rdf:about="http://www.miProyecto.owl#TieneReferenciaAAnalisis"/>
    </owl:onProperty>
   </owl:Restriction>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="http://www.miProyecto.owl#Documentos_de_DiseÃ±o"/>
</owl:Class>
<owl:Class>
  <owl:unionOf rdf:parseType="Collection">
   <owl:Class rdf:about="http://www.miProyecto.owl#miDocumento"/>
   <owl:Class rdf:about="http://www.miProyecto.owl#Documentos_de_Analisis"/>
  </owl:unionOf>
</owl:Class>
<owl:Class rdf:about="http://www.miProyecto.owl#Riesgo">
  <rdfs:subClassOf rdf:resource="http://www.miProyecto.owl#Elementos"/>
</owl:Class>
<owl:Class rdf:about="http://www.miProyecto.owl#Proyecto"/>
<owl:Class rdf:about="http://www.miProyecto.owl#Termino">
  <rdfs:subClassOf rdf:resource="http://www.miProyecto.owl#Elementos"/>
</owl:Class>
<owl:Class rdf:about="http://www.miProyecto.owl#Lista_de_Riesgos">
  <owl:equivalentClass>
   <owl:Restriction>
    <owl:someValuesFrom rdf:resource="http://www.miProyecto.owl#Riesgo"/>
    <owl:onProperty>
      <owl:ObjectProperty rdf:about="http://www.miProyecto.owl#TieneReferenciaAElemento"/>
    </owl:onProperty>
   </owl:Restriction>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="http://www.miProyecto.owl#Documentos_de_Analisis"/>
</owl:Class>
<owl:Class rdf:about="http://www.miProyecto.owl#CasosDeUso">
  <owl:equivalentClass>
```

```
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="http://www.miProyecto.owl#TieneReferenciaAElemento"/>
     </owl:onProperty>
     <owl:someValuesFrom>
      <owl:Class rdf:about="http://www.miProyecto.owl#Requerimiento_Funcional"/>
     </owl:someValuesFrom>
    </owl:Restriction>
   </owl:equivalentClass>
   <rdfs:subClassOf rdf:resource="http://www.miProyecto.owl#Documentos_de_Analisis"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.miProyecto.owl#Requerimiento_Funcional">
   <rdfs:subClassOf rdf:resource="http://www.miProyecto.owl#Elementos"/>
  </owl:Class>
  <owl:Class rdf:about="http://www.miProyecto.owl#Analista">
   <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.miProyecto.owl#Persona"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="http://www.miProyecto.owl#miDocumento"/>
    <owl:Class rdf:about="http://www.miProyecto.owl#Documentos_de_DiseÃ±o"/>
   </owl:unionOf>
  </owl:Class>
  <owl:Class rdf:about="http://www.miProyecto.owl#Arquitecto_de_Software">
   <rdfs:subClassOf rdf:resource="http://www.miProyecto.owl#Persona"/>
  </owl:Class>
  <owl:ObjectProperty rdf:about="http://www.miProyecto.owl#TieneReferenciaAAnalisis">
   <rdfs:range rdf:resource="http://www.miProyecto.owl#Documentos_de_Analisis"/>
   <rdfs:domain>
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="http://www.miProyecto.owl#Documentos_de_DiseÃ±o"/>
      <owl:Class rdf:about="http://www.miProyecto.owl#miDocumento"/>
```

```
    </owl:unionOf>
   </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.miProyecto.owl#TieneAutor">
  <rdfs:domain>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="http://www.miProyecto.owl#Documentos_de_Analisis"/>
     <owl:Class rdf:about="http://www.miProyecto.owl#Documentos_de_DiseÃ±o"/>
     <owl:Class rdf:about="http://www.miProyecto.owl#miDocumento"/>
    </owl:unionOf>
   </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.miProyecto.owl#Persona"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.miProyecto.owl#PerteneceAProyecto">
  <rdfs:range rdf:resource="http://www.miProyecto.owl#Proyecto"/>
  <rdfs:domain>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="http://www.miProyecto.owl#Elementos"/>
     <owl:Class rdf:about="http://www.miProyecto.owl#Persona"/>
     <owl:Class rdf:about="http://www.miProyecto.owl#miDocumento"/>
    </owl:unionOf>
   </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.miProyecto.owl#TieneReferenciaAElemento">
  <rdfs:domain>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="http://www.miProyecto.owl#Documentos_de_Analisis"/>
     <owl:Class rdf:about="http://www.miProyecto.owl#miDocumento"/>
    </owl:unionOf>
```

```
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.miProyecto.owl#Elementos"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:about="http://www.miProyecto.owl#TieneDescripcion">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.miProyecto.owl#Elementos"/>
        <owl:Class rdf:about="http://www.miProyecto.owl#miDocumento"/>
        <owl:Class rdf:about="http://www.miProyecto.owl#Persona"/>
        <owl:Class rdf:about="http://www.miProyecto.owl#Proyecto"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<Requerimiento_Funcional rdf:about="http://www.miProyecto.owl#RQ6">
  <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Mantener una lista de vuelos a distintas ciudades origen o destino</TieneDescripcion>
</Requerimiento_Funcional>
<Termino rdf:about="http://www.miProyecto.owl#T4">
  <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Destino</TieneDescripcion>
</Termino>
<Analista rdf:about="http://www.miProyecto.owl#An3">
  <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Mario</TieneDescripcion>
</Analista>
<Requerimiento_Funcional rdf:about="http://www.miProyecto.owl#RQ3">
  <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Vender Boletos a pasajeros registrados</TieneDescripcion>
</Requerimiento_Funcional>
<Termino rdf:about="http://www.miProyecto.owl#T8">
  <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
 >Vuelo</TieneDescripcion>
</Termino>
<Arquitecto_de_Software rdf:about="http://www.miProyecto.owl#Ar2">
 <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Jorge</TieneDescripcion>
</Arquitecto_de_Software>
<Termino rdf:about="http://www.miProyecto.owl#T2">
 <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Azafata</TieneDescripcion>
</Termino>
<Requerimiento_Funcional rdf:about="http://www.miProyecto.owl#RQ2">
 <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Ingresar datos de los pasajeros</TieneDescripcion>
</Requerimiento_Funcional>
<Requerimiento_Funcional rdf:about="http://www.miProyecto.owl#RQ7">
 <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Asignar tripulacion a vuelos</TieneDescripcion>
</Requerimiento_Funcional>
<Termino rdf:about="http://www.miProyecto.owl#T3">
 <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Boletos</TieneDescripcion>
</Termino>
<Analista rdf:about="http://www.miProyecto.owl#An1">
 <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Emilio</TieneDescripcion>
</Analista>
<Analista rdf:about="http://www.miProyecto.owl#An2">
 <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Jose</TieneDescripcion>
</Analista>
<Termino rdf:about="http://www.miProyecto.owl#T7">
 <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Piloto</TieneDescripcion>
</Termino>
<Arquitecto_de_Software rdf:about="http://www.miProyecto.owl#Ar1">
```

<TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

>Antonio</TieneDescripcion>

</Arquitecto_de_Software>

<Termino rdf:about="http://www.miProyecto.owl#T1">

<TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

>Asiento</TieneDescripcion>

</Termino>

<Riesgo rdf:about="http://www.miProyecto.owl#R2">

<TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

>Falta de aprobaciÃ³n para el proceso que determina la cantidad de millas a asignar</TieneDescripcion>

</Riesgo>

<Requerimiento_Funcional rdf:about="http://www.miProyecto.owl#RQ1">

<TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

>Mantener una lista de paises origen y destino</TieneDescripcion>

</Requerimiento_Funcional>

<Requerimiento_Funcional rdf:about="http://www.miProyecto.owl#RQ5">

<TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

>Mantener una lista de pilotos y azafatas</TieneDescripcion>

</Requerimiento_Funcional>

<Termino rdf:about="http://www.miProyecto.owl#T6">

<TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

>Pasajero</TieneDescripcion>

</Termino>

<Proyecto rdf:about="http://www.miProyecto.owl#P01">

<TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

>AerolÃneas del Sur</TieneDescripcion>

</Proyecto>

<Riesgo rdf:about="http://www.miProyecto.owl#R1">

<TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

>Ausencia del procedimiento para la Asignacion de asientos</TieneDescripcion>

</Riesgo>

<Requerimiento_Funcional rdf:about="http://www.miProyecto.owl#RQ4">

<TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

>Agregar millas de regalo en la venta de boletos</TieneDescripcion>

```
  </Requerimiento_Funcional>
  <Analista rdf:about="http://www.miProyecto.owl#An4">
    <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Sandra</TieneDescripcion>
  </Analista>
  <Termino rdf:about="http://www.miProyecto.owl#T5">
    <TieneDescripcion rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Origen</TieneDescripcion>
  </Termino>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.4.4, Build 579)  http://protege.stanford.edu -->
```

Anexo B Herramientas para el manejo de Ontologías

Para el manejo de ontologías en aplicaciones Java, se usó las siguientes herramientas:

- JENA [22], es un conjunto de herramientas orientadas a la creación de aplicaciones para la web semántica, desarrollado en JAVA por HP Labs. Jena puede trabajar indistintamente con una ontología creada en OWL o RDF (también funciona para ontologías escritas en DAM OIL). Esto es posible gracias al uso de un profile que permite la representación de las diferencias entre los distintos leguajes ontológicos. Los profiles se enlazan a una clase llamada OntModel que es una extensión de la clase Model de Jena. La clase Model permite el acceso a las sentencias en formato de recursos RDF, en tanto que OntModel agrega el soporte para las sentencias que conforman OWL, como clases y jerarquías de clases y propiedades e individuos. Con el uso de JENA la ontología queda disponible para su uso en aplicaciones
- Pellet [22], es un razonador OWL DL para JAVA que puede ser empleado en conjunto con JENA. Provee funcionalidad relacionada a la prueba de consistencia de una ontología, clasificación taxonómica, verificación de vinculaciones y respuestas a consultas SPARQL.
- Protégé [23], es un open source de libre distribución para la edición de ontologías. Permite la generación de proyectos para el manejo de conceptos y propiedades ontológicas, facilita además el uso de razonadores como el Pellet y genera archivos de la ontología en diversos formatos. Es el producto de la investigación de la Universidad de Standford.

Anexo C Instancias

A continuación se presenta las instancias creadas para las pruebas del prototipo:

<table>
<tr><td rowspan="4">Proyecto</td><td rowspan="4">Aerolíneas del Sur</td><td rowspan="4">Etapas</td><td rowspan="3">Análisis</td><td></td><td></td></tr>
<tr><td rowspan="2">Componentes</td><td>Elicitación</td></tr>
<tr><td>Especificación</td></tr>
<tr><td>Diseño</td><td>Componentes</td><td>Diseño</td></tr>
</table>

Elicitación	**ProductosComponente Inputs**	Lista de usuarios referentes
		Documento de justificación del proyecto
	ProductosComponente Outputs	Lista de requerimientos del proyecto
		Glosario
	Equipo Asignado	Emilio
		José
Especificación	**ProductosComponente Inputs**	Lista de requerimientos del proyecto
		Glosario
	ProductosComponente Outputs	CU 1. Manejar ABM de Destinos
		CU 2. Manejar ABM de Orígenes
		CU 3. Manejar ABM de Pilotos
		CU 4. Manejar ABM de Vuelos
		CU 5. Manejar ABM de Azafatas
		CU 6. Vender Boletos
		CU 7. Manejar ABM de Pasajeros
	Equipo Asignado	Sandra
		Mario
Diseño	**ProductosComponente Inputs**	CU 1. Manejar ABM de Destinos
		CU 2. Manejar ABM de Orígenes
		CU 3. Manejar ABM de Pilotos
		CU 4. Manejar ABM de Vuelos
		CU 5. Manejar ABM de Azafatas
		CU 6. Vender Boletos
		CU 7. Manejar ABM de Pasajeros
	ProductosComponente Outputs	RCU 1. Manejar ABM de Destinos
		RCU 2. Manejar ABM de Orígenes
		RCU 3. Manejar ABM de Pilotos
		RCU 4. Manejar ABM de Vuelos
		RCU 5. Manejar ABM de Azafatas
		RCU 6. Vender Boletos
		RCU 7. Manejar ABM de Pasajeros
	Equipo Asignado	Antonio
		Jorge