

Universidad Nacional De La Plata (UNLP)  
Facultad De Informática  
Centro Internacional Franco Argentino de Ciencias de la  
Información y Sistemas (CIFASIS)  
CONICET-UNICAM III-UNR

## Tesis Doctoral

# Contratos sensibles al contexto para el Dispositivo Hipertextual Dinámico

Alejandro R. Sartorio

**Director:** Dro. Gustavo Rossi  
**Co-Director:** Dra. Patricia San Martín  
**Asesor Científico:** Ms. Maximiliano Critia

**Miembros del jurado:** Jurado 1  
Jurado 2  
Jurado 3

Tesis presentada en la Facultad de Informática de La Plata, en  
cumplimiento parcial de los requisitos para optar al título de:

## Doctor en Informática

FECHA DE PRESENTACIÓN

Certifico que el trabajo incluido en esta tesis es el resultado de tareas de investigación originales y que no ha sido presentado para optar a un título de postgrado en ninguna otra Universidad o Institución.

XXXXX  
XXXXX

---

# Índice general

---

<b>1. Introducción</b>	<b>9</b>
1.1. Motivación . . . . .	4
1.2. Solución propuesta y contribución . . . . .	10
1.2.1. Primera noción de los elementos intervinientes en la solución	12
1.3. Limitaciones . . . . .	13
1.4. Organización del documento . . . . .	14
1.5. Publicaciones de Apoyo . . . . .	14
<b>2. Dispositivo Hipermedial Dinámico: Marco General</b>	<b>19</b>
2.1. Introducción2 . . . . .	19
2.1.1. El plano de las interacciones del DHD . . . . .	20
Las interacciones . . . . .	21
2.1.2. El plano sistémico del DHD . . . . .	23
Dinamismo sistémico del DHD . . . . .	24
2.1.3. El plano de la Ingeniería Computacional del DHD . . . . .	25
Campo teórico-metodológico interdisciplinario del DHD . .	25
2.2. Requerimientos DHD . . . . .	30
2.2.1. Breves antecedentes sobre el tratamiento de requerimientos 2	31
Captura de requisitos para los DHD . . . . .	32
2.2.2. Abstracción para el refinamiento de los RequerimientosDHD	34
2.3. Conclusiones . . . . .	36
<b>3. Estado del Arte</b>	<b>37</b>
3.1. Introducción . . . . .	37
3.2. El contexto en el DHD . . . . .	37
3.2.1. Caracterización del Contexto . . . . .	38
Origen del contexto en el DHD . . . . .	38
Modelado del contexto . . . . .	41
ContextoDHD: Contexto para los servicios DHD . . . . .	41
Fuentes de la información de contexto . . . . .	43
Contexto para los servicios de reconfiguración . . .	43
Modelado de la sensibilidad del contexto . . . . .	43
Mecanismos de Envolturas . . . . .	44
Mecanismos de Ejecución . . . . .	44
3.3. Sistemas Sensibles al Contexto . . . . .	45

3.3.1.	Propuestas para la Identificación del Contexto . . . . .	48
3.3.2.	Visiones en la perspectiva histórica . . . . .	50
	Vannevar Bush's Memex (1945) . . . . .	50
	Alan Kay's Dynabook (circa 1977) . . . . .	50
	Mark Weiser; "The Computer for the 21st Century (1991) . . . . .	51
	Bill Schillit: "Context-Aware Computing Applica- tions" (1994) . . . . .	51
	Don Norman: "The invisible Computer" (1998) . .	52
3.4.	Componentes bases para los Sistemas Sensibles al Contexto . . . .	52
3.4.1.	Context Toolkit . . . . .	52
3.4.2.	Proyecto UWA . . . . .	55
	Goal . . . . .	56
	Service . . . . .	56
	Environment . . . . .	57
	Context . . . . .	58
	Profiles . . . . .	58
	Requirements . . . . .	58
	Application . . . . .	58
	Customisation Rules . . . . .	59
	The Meta Level . . . . .	59
3.5.	Hacia la Ingeniería de Software . . . . .	60
3.6.	Conclusiones . . . . .	61
<b>4.</b>	<b>ArqDHD: Arquitectura de Software para los Dispositivos Hiperme- diales Dinámicos</b>	<b>63</b>
4.1.	Introducción . . . . .	63
4.2.	Conceptos Fundamentales . . . . .	66
4.2.1.	ComponenteDHD . . . . .	66
4.2.2.	ConectorDHD . . . . .	67
4.2.3.	PuertoDHD . . . . .	70
4.3.	Visión arquitectónica de los DHD . . . . .	71
4.3.1.	Propuestas de Arquitectura del Software para los DHD . .	71
4.3.2.	Propuestas basadas en el Desarrollo Dirigido por Modelos para Aplicaciones Web . . . . .	74
4.3.3.	Situación de ArqDHD en la Investigación Actual . . . . .	77
	Estilo Arquitectónico . . . . .	78
4.3.4.	El contrato como conector . . . . .	79
	Perspectiva de modelado . . . . .	82
4.3.5.	Los conectores en entornos adaptativos . . . . .	83
4.3.6.	Tipos de Dinamismo . . . . .	84
4.3.7.	Reconfiguración Dinámica . . . . .	85
4.3.8.	Tipos de Reconfiguración . . . . .	86
4.3.9.	Operaciones Básicas de Reconfiguración . . . . .	87
4.4.	Descripción arquitectónica de los DHD . . . . .	88
4.4.1.	Darwin . . . . .	88
4.4.2.	Wright Dinámico . . . . .	92
4.5.	Conclusiones . . . . .	93

<b>5. Contratos sensibles al contexto</b>	<b>95</b>
5.1. Introducción . . . . .	95
5.2. Hacia la definición de ContratoDHD . . . . .	95
5.2.1. Características que debe cumplir el ContratoDHD . . . . .	95
5.3. Hacia la definición de contratos para el DHD . . . . .	98
5.4. ContratosDHD: Contratos sensibles al contexto . . . . .	98
Sistema que proveen soportes para el desarrollo de software basado en contratos . . . . .	99
5.4.1. Componentes esenciales de los contratos sensibles al contexto	102
5.4.2. Una implementación del subsistema para CEC#1 . . . . .	105
Infraestructura para serviciosDHD . . . . .	106
5.4.3. Coordinación de los contratos sensibles al contexto . . . . .	106
Coordinación . . . . .	108
5.5. Condicionales especiales para las reglas de los contratos . . . . .	109
5.5.1. MCondicionales . . . . .	109
Modelo de integración . . . . .	111
5.5.2. DMCondicionales . . . . .	112
Detalles del modelo de integración . . . . .	116
Mecanismo de comunicación . . . . .	117
Caso de uso . . . . .	118
Hacia la implementación . . . . .	119
5.6. Conclusiones . . . . .	120
<b>6. Framework para la inyección de contratos sensibles al contexto en entornos Web colaborativos</b>	<b>121</b>
6.1. Introducción . . . . .	121
6.2. Contratos con características sensibles al contexto . . . . .	123
6.2.1. Elementos de los contratos sensibles al contexto . . . . .	124
6.3. Modelo de integración de Sakai con contratos . . . . .	126
6.4. Implementación de contratos en Sakai . . . . .	127
6.4.1. Niveles de flexibilidad de Sakai . . . . .	127
6.4.2. Patrones de diseño para la coordinación de contratos sen- sible al contexto . . . . .	128
6.5. Caso de estudio para la implementación de contratos . . . . .	130
Fragmentos del primer archivo - (archivo 1) . . . . .	131
Fragmentos del segundo archivo - (archivo 2) . . . . .	132
6.6. Conclusión . . . . .	133
<b>7. Implementaciones del framework DHD: Casos de usos</b>	<b>135</b>
7.1. Usos de los contratos sensibles al contexto: Evidencias en el uso de entornos E-learning . . . . .	135
7.1.1. Contratos context-Aware para transacciones en los DHD . . . . .	137
Elementos de la componente contrato . . . . .	137
7.1.2. Documentación de los DHD procesos . . . . .	139
7.1.3. UWATc+: una adaptación de UWAT+ para el modelado de DHD procesos . . . . .	141
Requerimientos para el diseño de DHD Transacciones . . . . .	142
7.1.4. Diseño de procesos e-learning con UWATc+ . . . . .	143
Determinación de Requerimientos . . . . .	144
Diseño de DHD-Transacciones . . . . .	145

Diagrama de un Contrato . . . . .	147
7.2. Desarrollos Context Aware en el campo del sonido . . . . .	149
7.2.1. Conclusión . . . . .	151
7.3. DHD: Prototipo experimental . . . . .	151
7.4. Intervención del DHDCca en un modelo de simulación DEVS para DHD . . . . .	151
7.4.1. Construcción de métricas para el DHD . . . . .	151
7.4.2. Integración en el modelo DEVS . . . . .	155
Condicionales para la coordinación de contratos . . . . .	158
Modelo conceptual de integración . . . . .	159
Implementación en un caso de uso . . . . .	161
7.4.3. Consideraciones generales . . . . .	165
<b>8. Otras implementaciones de los ContratosDHD</b>	<b>167</b>
8.1. Condicionales DEVS en la coordinación de contratos sensibles al contexto para los DHD . . . . .	167
8.1.1. Introducción . . . . .	167
8.1.2. Aspecto tecnológico de los DHD . . . . .	169
Condicionales para la coordinación de contratos . . . . .	172
8.1.3. Modelo conceptual de integración . . . . .	173
8.1.4. Implementación en un caso de uso . . . . .	176
Representación de los Condicionales DEVS a través de UWATc . . . . .	176
Ejemplo de ejecución de la métrica en PowerDEVS . . . . .	178
8.1.5. Conclusiones . . . . .	180
8.2. SEPI: una herramienta para el Seguimiento y Evaluación de Pro- cesos Interactivos del DHD . . . . .	180
8.2.1. Introducción . . . . .	180
8.2.2. Elementos para la integración . . . . .	182
8.2.3. Modelo conceptual de integración . . . . .	184
8.2.4. Implementación en un caso de uso . . . . .	186
Paso 1: Escritura y lectura del XML . . . . .	186
Paso 2: Acceso a la base de datos . . . . .	187
Paso 3: Configuración de coeficientes de la métrica . . . . .	188
Paso 4: Ejecución de la simulación . . . . .	188
Paso 5: Lectura de resultados . . . . .	189
Paso 6: Insertar valores en los contratos . . . . .	189
8.2.5. Conclusiones . . . . .	189
<b>9. Conclusiones y trabajos Futuros</b>	<b>193</b>
9.1. Conclusiones . . . . .	193
9.2. Lineas de trabajo futuras . . . . .	193
9.2.1. Propuesta de actividades . . . . .	196
9.2.2. Factibilidad . . . . .	196

---





# CAPÍTULO 1

---

## Introducción

---

## **Resumen**

Resumen para la gilada

---

Esta tesis se inició en el 2005 atendiendo a requerimientos de mayor flexibilidad y dinamismo de los entornos e-learning, planteados por un grupo interdisciplinario de I+D del CONICET-UNR dedicado a problemáticas de integración de las Tecnologías de la Información y Comunicación (TIC) en contextos organizacionales de Educación Superior. Dichos requerimientos se enfocaban hacia mejorar la calidad y posibilidades de desarrollo de procesos educativos, de investigación y de producción hipermedial bajo la modalidad de taller físico-virtual. Esta necesidad permitió la elaboración de las primeras hipótesis de la presente investigación delimitando los alcances del estudio circunscriptos a la Ingeniería de Software y Ciencias de la Computación aplicada a la interpretación de la complejidad de estos requerimientos y su correspondiente infraestructura de resolución.

El desafío que afrontó este trabajo en todo su trayecto, fue colaborar desde las Ciencias de la Computación al despliegue del diálogo y producción interdisciplinar del grupo de trabajo sobre el campo conceptual referido a la integración de TIC en procesos antes mencionados. Así, se fue generando una construcción teórica y metodológica denominada consensuadamente como "Dispositivo Hipermedial Dinámico DHD-(P. San Martín, et. al, 2008). Desde la experiencia previa del autor de este trabajo, los requerimientos planteados demandaron realizar un estudio específico de proyectos centrados especialmente en el campo del "e-learning".

De estas indagaciones se adhirió a que los sistemas colaborativos se mostraban como una propuesta racional para sostener modelos donde se pueden implementar soluciones tecnológicas para requerimientos funcionales que deriven de un requisito de más alto nivel en los que aumenta el grado de subjetividad. Esto se presenta debido a que es necesario tener en cuenta cuestiones de contexto, tipos de relaciones y vínculos que dan lugar a diferentes niveles de complejidad y de interpretación.

Actualmente se evidencia un creciente interés por metodologías de trabajo más efectivas, que toman muy en cuenta el potencial de cada uno de los sujetos. Se busca generar conocimiento útil a partir de la información disponible y el desarrollo de competencias que permitan aplicar ese conocimiento efectivamente. No obstante, la complejidad de muchos problemas requiere del trabajo de grupos, constituidos por personas capaces de interactuar de forma sinérgica para buscar soluciones adecuadas y con alto grado de creatividad (1). La construcción de conocimiento aplicando metodologías de diálogo interdisciplinar ha dejado de ser una opción para convertirse en una necesidad. Es en este sentido que se sostiene en esta tesis una perspectiva computacional acorde al contexto, totalmente integrada a los postulados del Programa de Investigación, Desarrollo y Transferencia "Dispositivos Hipermediales Dinámicos" radicado en el CIFASIS (CONICET-UNT-UPCAM) bajo la dirección de la Dra. Patricia San Martín.

Debido al papel protagónico de las TIC en la actual Sociedad de la Información y del Conocimiento y, el vertiginoso desarrollo que han tenido en los últimos años, se han planteado paradigmas y esquemas de trabajo que integran a las ciencias computacionales con otros campos del conocimiento tales como las ciencias humanas y sociales, las ciencias administrativas, el diseño gráfico y artístico, etc., en un intento de construcción de un nuevo contexto físico-virtual pero reconociendo al tiempo, que pese al potencial de las TIC, gran parte de los problemas a resol-

---

ver, no pueden ser abordados exclusivamente desde una perspectiva tecnológica (18; 3).

La elección de metodologías y paradigmas en los que se integran tecnologías y funcionalidades conceptuales, necesitan ser interpretados como posibles instancias que se pueden configurar. Por lo tanto, se pueden interpretar propiedades generales de los sistemas a partir de las posibles configuraciones de un estado determinado, de un caso de uso concreto. Este análisis es tratado en el grupo de I+D “Obra Abierta: DHD para educar e investigar” a través de la teoría de los sistemas complejos y se tiene en cuenta como ítems para la clasificación de aplicaciones “Groupware”.

Los aspectos que se tendrán en cuenta en esta tesis sobre los sistemas colaborativos son:

Nivel de automatización: Las aplicaciones “Groupware” como considera Patricia Schnaidt, se pueden clasificar de acuerdo al tipo de trabajo en grupo que apoyan: (8)

*Flujo de documentos, Automatización de procesos, Automatización de tareas, Herramientas flexibles de trabajo en grupo.*

- De acuerdo al espacio-tiempo de los miembros del grupo: Se consideran las aplicaciones “groupware” de acuerdo al tipo de interacción de los miembros del grupo de trabajo (6).
  - Interacción Sincrónica
  - Interacción Asincrónica
  - Distribución Sincrónica
- De acuerdo al manejo de información. Según el soporte que brindan (9) los sistemas “groupware” pueden clasificarse en:
  - Sistemas para compartir Información
  - Sistemas cooperativos.
  - Sistemas concurrentes.
- De acuerdo al propósito de la aplicación: es posible encontrar dos tipos de aplicaciones dependiendo de su propósito.
  - Aplicaciones de propósito general.
  - Aplicaciones de propósito específico.
- De acuerdo al tipo de aplicación: Dependiendo del tipo de aplicación y funcionalidad se pueden clasificar en los siguientes grupos:
  - Sistemas de mensajes por computador
  - GDSSs (Group Decision Support Systems).

## 1.1. Motivación

---

- Sistemas de coordinación: Orientados a la formas, orientados al proceso, orientados al diálogo, orientados a la estructura de comunicación.
- De acuerdo al tipo de reconfiguración orientada a la Interactividad DHD(96).  
Dependiendo del grado de expresión que se tiene para reconfigurar el sistema teniendo en cuenta información procesada para la representación de contexto interno y del entorno.
  - Reconfiguración dinámica.
  - Adaptabilidad.
  - Sensibilidad al contexto.
  - Sistemas expertos: bases de conocimientos y motores de inferencias.

Los aportes de originales de la tesis se fundamentan en publicaciones de circulación internacional acreditadas proponiendo una perspectiva innovadora referida a la reconfiguración dinámica de los entornos colaborativos orientada a la Interactividad DHD.

## 1.1. Motivación

A medida que el avance en la investigación y desarrollo de entornos colaborativos brindan mejoras e innovaciones en herramientas (videoconferencias, portfolios, wikis, workshops, etc.) y sus respectivos servicios, crece la cantidad de posibles configuraciones de los espacios colaborativos.

Estas configuraciones abarcan diferentes tipos de requerimientos pertenecientes a las etapas de diseño, desarrollo e incluso exigen que el espacio colaborativo se "adapte" en tiempo de ejecución. A partir de estos requerimientos iniciales se definen los procesos e-colaborativos (denominados en esta tesis como Pe-colaborativos) (10) de manera semejante a procesos de negocio en otros dominios de aplicación. Esta denominación parte de la evolución del recorrido teórico iniciado a partir de las transacciones Web, que dieron lugar a teorizar el concepto de transacciones e-learning (59) y que en el trayecto de desarrollo de esta investigación evolucionaron finalmente hacia el concepto de procesos e-colaborativos.

Al igual que los procesos de negocios en una Aplicación Web convencional, los Pe-colaborativos están compuestos por transacciones Web (10). En este contexto, una transacción (o transacción e-colaborativa) es definida como una secuencia de actividades que un usuario ejecuta a través de una Aplicación colaborativa con el propósito de efectuar una tarea o concretar un objetivo, donde el conjunto de actividades, sus propiedades y las reglas que controlan sus ejecuciones dependen del Pe-colaborativos que la Aplicación debe brindar. Un ejemplo de esto es la posibilidad de implementar una estrategia didáctica que le brinde a un alumno que participa en un entorno virtual colaborativo la posibilidad de acceder a un tipo de Objeto Digital Educativo, dependiendo de sus intervenciones en los Foros.

Estos requerimientos resultan difíciles de implementar en las actuales aplicaciones e-learning de extendido uso a nivel global.

Las características funcionales de aplicaciones Web colaborativas (AW-Colaborativas), como Sakai <sup>a</sup>, se basan en brindar navegación entre páginas a través de links y ejecución de transacciones colaborativas desde las herramientas (ej., Foros, Anuncios, Exámenes, Blogs, etc.) que utilizan los servicios de la plataforma (ej., edición, manejo de audio y vídeo, consultas, navegación, etc.).

En el marco de los análisis efectuados se observó que el proyecto Sakai brinda una de las propuestas más consolidadas de diseño y desarrollo de entornos colaborativos, teniendo en cuenta los anteriores items 1.1 analizados.

Sakai está orientado a herramientas que se implementan a través de servicios comunes (servicios bases). Por ejemplo, el servicio de edición de mensajes es utilizado en las herramientas Foro, Anuncio, Blog, PorFolio, etc. Más aun, otras de las características salientes de Sakai es la versatilidad para su extensión y/o configuración que permite alterar ciertas configuraciones en tiempo de ejecución, por ejemplo, instrumentar una nueva funcionalidad en un servicio base de Sakai.

Sin embargo, estas soluciones no pueden resolver aquellos Pe-Colaborativos que involucren cambios en el comportamiento de la relaciones entre un componente (cliente) que ocasiona, a través de un pedido, la ejecución de un componente servidor (proveedor). Estos cambios refieren a la capacidad de adaptación dinámica del sistema (11) extendiendo, personalizando y mejorando los servicios sin la necesidad de recompilar y/o reiniciar el sistema.

Esta tesis doctoral presenta una propuesta para la incorporación (agregado) de propiedades de adaptación dinámicas, reconfiguración e inteligencia (AdRI) a los servicios bases del framework Sakai, especialmente diseñadas para implementar Pe-Colaborativos definidos en el marco de (12) donde se requieren nuevos aspectos de adaptación (61) implementados a partir del framework Sakai. Dicha implementación se ajusta a los requisitos y requerimientos (capítulo ??) del Dispositivo Hipermedial Dinámico, que como red sociotécnica responsable posibilita a los sujetos desarrollar actividades académicas y profesionales bajo la modalidad de taller físico-virtual.

Cuando se refiere a las propiedades de inteligencia de un framework, se atribuye a un entorno con características heterogéneas con numerosos componentes de software y hardware (4). Esta diversidad implica ciertos desafíos problemáticos que demandan una atención especial en los diseños solicitando una preparación para usos complejos multisensoriales:

- Se han de integrar y gestionar distintos tipos de tecnología, con el consecuente aumento de complejidad en el desarrollo. El usuario puede utilizar múltiples modos (habla, gestos, tacto, etc.) para interactuar con el en-

---

<sup>a</sup><http://sakaiproject.org/>

## 1.1. Motivación

---

torno, con lo que se requieren interfaces de usuario muy variadas. Por otro lado la distribución de la información requiere distintos tipos de redes, según la naturaleza de ésta.

- Los componentes pueden estar altamente distribuidos. Tanto los sensores, que se encargan de recoger la información del entorno, como los actuadores, que transmiten la respuesta del entorno al usuario, pueden tener localizaciones distribuidas. Nuevamente se añade una dificultad extra a la hora de desarrollar aplicaciones.
- La configuración del entorno es dinámica. No se puede prever siempre el momento en que se conectan y desconectan nuevos dispositivos o entran y salen nuevos usuarios.
- El sistema tiene que estar funcionando siempre. Esto quiere decir que se deben evitar la mayor cantidad posibles de tareas en tiempo de compilación. En este sentido se ven afectados cuestiones de diseño, arquitectura e implementación que permitan la mayor versatilidad para la ejecución de cambios en tiempo de ejecución. Además, es importante tener en cuenta previamente, cuáles van a ser las componentes de los sistemas y qué lugares ocupan para prepararlas para el cambio o reconfiguración dinámicas.

La combinación de las aplicaciones sensibles al contexto y los entornos activos conlleva una serie de dificultades adicionales. A la información contextual generada por usuarios, dispositivos y aplicaciones hay que añadir el contexto del entorno.

Dentro de un entorno activo se pueden encontrar fuentes de información contextual de diversa naturaleza. Las fuentes pueden ser muy cercanas al mundo físico, o por el contrario pueden estar relacionadas con el mundo virtual. El primer conjunto se compone de toda clase de dispositivos ligados al entorno que interactúan con el mundo físico tales como sensores, conmutadores, electrodomésticos, pantallas, micrófonos, altavoces, etc. El segundo conjunto incluye aquellos componentes puramente computacionales, tales como gestores de diálogos, agentes inteligentes, clientes de correo, etc. Ya se ha mencionado la naturaleza distribuida de estos componentes, lo cual implica mayor complejidad en el desarrollo de aplicaciones.

Un problema importante surge por la diferencia en cuanto al nivel de abstracción en la información contextual aportada por los distintos componentes. Los sensores manejan información de alta resolución que es rica en detalles pero pobre en cuanto a abstracción. En cambio las aplicaciones pueden requerir información contextual más elaborada que la que aportan los sensores. Los desarrolladores tienen que convivir con información que tiene distinta resolución, lo que implica distintos formatos y distintas redes de distribución.

Finalmente, la interfaz de comunicación con el usuario debe ser lo más flexible posible, para promover la interactividad tanto en los aspectos comunicacionales como de acceso y edición de información que permiten actualmente las TIC. El logro de una integración natural de estas tecnologías a las actividades cotidianas

de los usuarios es un fin representativo de la posibilidad de construcción de un nuevo contexto físico-virtual (capítulo ??).

Retomando aspectos de I+D, es fundamental que la interfaz tenga el suficiente nivel de expresión para que se puedan instrumentar adecuadamente las reconfiguraciones dinámicas, ya que en ellas se encuentra la máxima complejidad funcional. En este punto el contexto también juega un papel importante, ya que es claro que la interacción sujeto-ordenador no es tan efectiva como la comunicación intersubjetiva donde se ponen en juego un sin número de posibilidades de interacción entre los sujetos y en relación al contexto.

En la mayoría de los casos resulta una adecuación forzada del sujeto a la máquina, solución contrapuesta a lo que se espera de una tecnología interactiva.

Para subsanar esta contradicción, se ha señalado que un punto importante es determinar cuál es la información contextual que el usuario debe suministrar al sistema para conseguir una mejor interacción. Entonces, una vez determinado el contexto, es preciso capturarlo. Esto requiere de múltiples sensores e interfaces de tal forma que se pueda captar tanto la interacción explícita como la implícita del usuario, teniendo en cuenta que este proceso se realice de forma no intrusiva.

Tal como se deduce del título <sup>b</sup> el planteamiento de esta tesis tiene en cuenta el contexto como una componente de primera clase.

Ahora bien, el término información contextual tiene múltiples acepciones que dependen del dominio de aplicación. Incluso si se restringe éste al campo de las Ciencias de la Computación, se registra aún un amplio espectro de posibles definiciones. En la presente tesis se considera el contexto en el mismo sentido que le dan los trabajos realizados en aplicaciones sensibles al contexto. Esta área engloba aplicaciones y dispositivos que consideran como una entrada más del sistema la información sobre las circunstancias bajo las cuales operan. Estas circunstancias pueden ser la localización, la tarea que está realizando el usuario, otros recursos que se encuentren cercanamente, las condiciones ambientales del entorno, etc. La definición y representación del contexto es un punto central de este trabajo que se tratará en profundidad en el capítulo ?? teniendo en cuenta que el contexto para el DHD necesita un tratamiento específico.

El alcance del estudio queda acotado a las aplicaciones sensibles al contexto que operan dentro de un entorno inteligente. Este consiste en una infraestructura restringida por unas barreras físicas y compartida por un conjunto de aplicaciones, dispositivos y sujetos. El adjetivo inteligente se emplea para indicar la habilidad de adquirir información de forma autónoma y de emplearla para adaptarse a las necesidades de los sujetos intervinientes. El entorno se convierte en una aplicación sensible al contexto más, contribuyendo activamente en la interacción con el usuario. Así, en la literatura también se pueden encontrar referencias a los entornos inteligentes como entornos activos ("Active Environment") o espacios activos ("Active Spaces"). Un hogar, una oficina, una clase, un vehículo o un restaurante

---

<sup>b</sup>Contratos Sensibles al Contexto para el Dispositivo Hipermedial Dinámico



## 1.1. Motivación

---

son ejemplos posibles de entornos activos.

Para situar la importancia del contexto y su relación con los entornos inteligentes es necesario referirse a la Computación Ubicua, una tercer área de investigación que engloba a los dos temas centrales de la tesis. La Computación Ubicua ??, también es conocida como Computación Pervasiva ("Pervasive Computing"). Ya Mark Weiser (1993), propone trasladar la capacidad de computación de los rígidos y voluminosos ordenadores personal de la época a miles de dispositivos diseminados por el entorno, de forma que las computadoras puedan fundirse con el entorno hasta volverse invisibles al usuario.

Esta perspectiva dan lugar a requerimientos centrados en el logro del máximo posible nivel de encapsulamiento para los procesos de reconfiguración dinámica y usos de servicios reconfigurados. O sea que los usuarios no deberían acceder a cómo están implementadas las reconfiguraciones y su habilitación para el uso. De alguna manera esto determina alcanzar un nivel de transparencia hacia el usuario que está mediado por el tipo de tecnología. En este sentido para alcanzar niveles óptimos de transparencia será necesario aún un salto tecnológico importante. Dicho según las palabras de Donald Norman (5):

Necesitamos movernos hacia la tercera generación de ordenadores personales, la generación donde la máquina desaparece de la vista, donde podemos volver a concentrarnos en las actividades y objetivos de nuestra vida.

(Donald Norman)

Hay tres transformaciones que devienen de la Computación Ubicua en las aplicaciones informáticas que también se deben tener en cuenta cómo condiciones a respetar. Estas transformaciones fundamentan las decisiones del uso de aplicaciones Web sobre las de escritorio.

- Reconocer el contexto.

Uno de los puntos débiles de las aplicaciones de escritorio es la insensibilidad ante los cambios de contexto del usuario. Mientras que se registra un considerable avance en la modelización del usuario y la adaptación según diferentes perfiles y preferencias, las aplicaciones tradicionales muestran el mismo comportamiento independientemente de la situación en que se encuentre el usuario. El contexto se ha reconocido como una parte fundamental de la comunicación humana (19), por lo que debe ser incorporado también en el diseño de los sistemas informáticos para acercarlos a los códigos humanos de comunicación. La localización, la actividad o el foco de atención del usuario, entre otras variables contextuales tienen que formar parte de las nuevas aplicaciones ubicuas.

- Actuar proactivamente.

El diálogo entre la aplicación y el usuario puede ser iniciado o bien por el usuario, o bien por la aplicación, o bien una mezcla de ambos. En general, la mayor parte de las aplicaciones de escritorio son pasivas, ya que es el usuario el que debe tomar la iniciativa. Un campo que ha prestado especial interés por la proactividad son los agentes personales. Éstos se encargan de buscar y mostrar información según la preferencias del usuario pero sin necesitar supervisión explícita. En el marco de la Computación Ubicua es preciso introducir en el diseño de las aplicaciones cierto grado de proactividad que permita liberar la atención del usuario cuando no sea imprescindible. Para ello se hace necesario realizar modelos del comportamiento humano (20) que permitan inferir las necesidades del usuario.

- Funcionalidades ubicuas.

Actualmente para que una aplicación sea ubicua es necesario instalarla en cada uno de los dispositivos donde se quiere emplear, teniendo en cuenta que para cada dispositivo se cargará una versión distinta que se ajuste a las restricciones que éste imponga. Esto incrementa la complejidad en el desarrollo, mantenimiento e interoperabilidad de las diferentes versiones de la aplicación. Una aproximación para solucionar este problema es mantener una funcionalidad única y generar la interfaz dinámicamente adaptándose a lo que requiera el usuario en cada momento (21). En este sentido se están realizando importantes esfuerzos por estandarizar lenguajes de representación de interfaces de usuario abstractas como UIML (22) o XIML (23), de modo que se pueda definir una interacción genérica independiente del modo a emplear y ligarla dinámicamente con la funcionalidad.

La consecución de estos objetivos recae sobre tres tecnologías: la computación ubicua, las aplicaciones sensibles al contexto y los entornos inteligentes. Recientemente, a la combinación de estas tres áreas se le ha denominado Inteligencia Ambiental<sup>c</sup>. Esta se refiere a la presencia de un entorno digital que es interactivo, sensible, y adaptativo a la presencia de sus ocupantes (22).

La Inteligencia Ambiental pretende fundir los conceptos de ubicuidad y transparencia en un diseño centrado en el usuario que dé como resultado un verdadero ordenador invisible. Para esta tesis, esta premisa es tenida en cuenta para la elección de propuestas tecnológicas externas utilizadas en las implementaciones. Esto quiere decir, que no se abordará la problemática de formar Ambientes Inteligentes a partir de dispositivos físicos. La intención se centra en tomar alguno de los fundamentos para aplicarlos en el tipo de propuesta de solución que se persigue. Se promueve el diseño centrado en el usuario y el uso de sistemas altamente interactivos. El objetivo final es conseguir que las tecnologías previamente mencionadas ayuden de manera no invasiva al desarrollo de procesos para educar, investigar y producir mediatizados por el DHD.

---

<sup>c</sup> Documento donde por primera vez se introduce el concepto y definición de "Inteligencia Ambiental" [http://www.epstein.org/brian/ambient\\_intelligence.htm](http://www.epstein.org/brian/ambient_intelligence.htm)

### 1.2. Solución propuesta y contribución

La propuesta de solución a los requerimientos mencionados sobre Adaptación Dinámica, Reconfiguración e Inteligencia (AdRI) está enfocada a la inyección de una nueva componente dentro de un sistema tecnológico origen. Esta componente a su vez está conectada con nuevos subsistemas que de forma estratificada se incorporan al original. Una posible representación de esta interpretación se describe en la figura 1.1 para colaborar en la comprensión sobre cuál es la estructura en la que se basa la propuesta de solución.

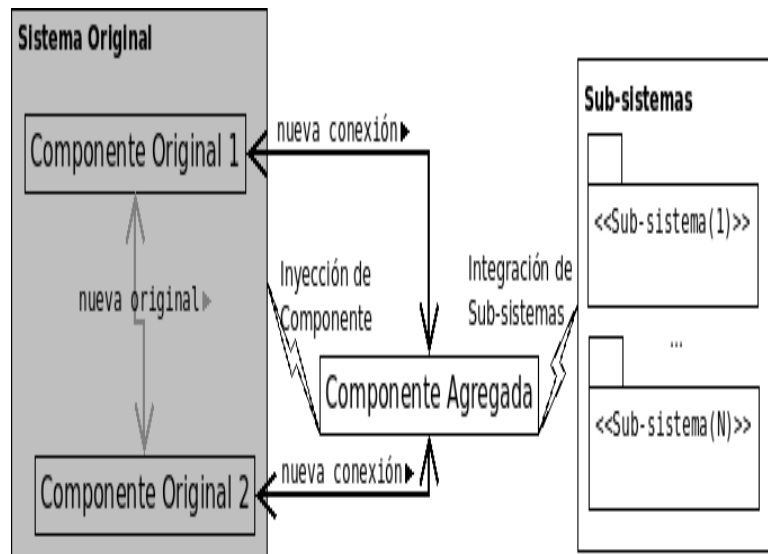


Figura 1.1: Estructura de la solución

La parte gris de la figura representa las componentes de un sistema original con sus componentes y relaciones. Luego se representa el agregado de una nueva componente (*componente agregada*) que se relaciona con las *componentes originales* del sistema por medio de *nuevas conexiones*. Esta nueva relación se denomina *inyección de componentes*. Luego la *componente agregada* se conecta con diferentes subsistemas. De esta manera queda conformado un nuevo sistema a partir del original, proponiéndose una "estructura de la solución para el DHD" (EstDHD).

En este caso, la componente agregada es el comienzo de la construcción de un modelo de contrato orientado a la implementación de servicios sensibles al contexto. El uso de contratos parte de la noción de Programación por Contrato ("Programming by Contract") de Meyer (16) basada en la metáfora de que un elemento de un sistema de software colabora con otro, manteniendo obligaciones y beneficios mutuos. En el dominio de aplicación cabe considerar que un objeto cliente y un objeto servidor "acuerdan" a través de un contrato (representado con un nuevo objeto) que el objeto servidor satisfaga el pedido del cliente, y al mismo tiempo el cliente cumpla con las condiciones impuestas por el proveedor.

Como ejemplo de la aplicación de la idea de Meyer en un dominio de sistema colaborativo e-learning se expone la situación en que un usuario (cliente) utiliza un servicio de edición de mensajes (servidor) a través de un contrato que garantizará las siguientes condiciones: el usuario debe poder editar aquellos mensajes que tiene autorización según su perfil (obligación del proveedor y beneficio del cliente); el proveedor debe tener acceso a la información del perfil del usuario (obligación del cliente y beneficio del proveedor).

A partir de la conceptualización de contratos según Meyer se propone una extensión por medio del agregado de nuevas componentes para instrumentar mecanismos que permitan ejecutar acciones dependiendo del contexto (figura 1.1).

En aplicaciones sensibles al contexto (15), el contexto (o información de contexto) es definido como la información que puede ser usada para caracterizar la situación de una entidad más allá de los atributos que la definen. En un caso del campo educativo, una entidad es un usuario (alumno, docente, etc.), lugar (aula, biblioteca, sala de consulta, etc.), recurso (impresora, fax, etc), u objeto (examen, trabajo práctico, etc.) que se comunica con otra entidad a través del contrato. En (18) se propone una especificación del concepto de contexto partiendo de las consideraciones de Dourish (24) y adaptadas al dominio de sistemas colaborativos con funcionalidades e-learning, que se consideran en este trabajo.

Contexto es todo tipo de información que pueda ser censada y procesada, a través de la aplicación (por ejemplo, una aplicación colaborativa e-learning), que caracterizan a un usuario o entorno, por ejemplo: intervenciones en los foros, promedios de notas, habilidades, niveles de conocimientos, máquinas (direcciones ip) conectadas, nivel de intervención en los foros, cantidad de usuarios conectados, fechas y horarios, estadísticas sobre cursos, etc.

En términos generales, la coordinación de contratos es una conexión establecida entre un grupo de objetos aunque en los límites de este trabajo sólo se consideran dos objetos: un cliente y un servidor.

Cuando un objeto cliente efectúa una llamada a un objeto servidor (ej., el servicio de edición de la herramienta Foro), el contrato "intercepta" la llamada y establece una nueva relación teniendo en cuenta el contexto del objeto cliente, el del objeto servidor, e información relevante adquirida y representada como contexto del entorno [20]. Como condición necesaria, el uso de contratos no debe alterar la funcionalidad de la implementación de los objetos participantes, aunque sí se espera que altere la funcionalidad del sistema.

A continuación se presenta sintéticamente un modelo conceptual de contratos sensibles al contexto de donde se infieren las distintas soluciones propuestas en esta tesis. Se brindarán detalles sobre algunos de los componentes y relaciones esenciales para la integración de este modelo con el framework colaborativo que se identifica con la parte gris de la figura 1.1 y algunos de sus sub-sistemas representados.

## 1.2. Solución propuesta y contribución

### 1.2.1. Primera noción de los elementos intervinientes en la solución

Un contrato que siga las ideas de Meyer contiene toda la información sobre los servicios que utilizarán los clientes. Para incorporar sensibilidad al contexto los contratos deberán tener referencias sobre algún tipo de información de contexto que posibilite su utilización.

En el diagrama de relaciones entre entidades analizado en la Figura 1.2 se presenta una primera descripción de los elementos que componen el concepto de contrato sensible al contexto. A lo largo de la tesis este mismo diagrama será utilizado para describir diferentes aspectos de una misma solución.

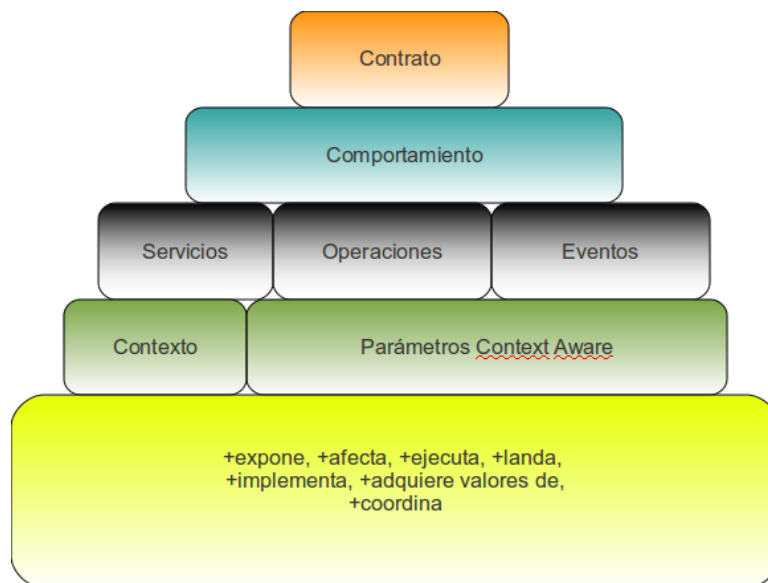


Figura 1.2: PirámideDHD: Primera interpretación de contratos

La configuración de este diagrama se centra en las etapas de diseño e implementación de diferentes modelos de integración con sub-sistemas (capítulos ??).

En la figura se muestran los elementos tenidos en cuenta para la instrumentación de la noción de contrato. Cada uno de los bloques apilados (desde el pilar 1 al pilar 5) en forma de piramidal pueden representar componentes abstractas o concretas. En el bloque base se encuentran las relaciones que intervienen entre las componentes superiores (pilar 1). Luego aparecen los bloques de contexto y parámetros context-aware (pilar 2), que permitirán configurar relaciones que posibilitan resolver requerimientos de adaptación. Seguidamente se encuentran los bloques que identifican componentes concretas que integran las herramientas colaborativas (pilar 3) que están sujetas a modificaciones para establecer las relaciones del bloque base para que se incorporen los elementos intermedios (pilar 2). Los últimos dos pilares tienen que ver con el comportamiento (pilar 4) que implementan los pilares inferiores, el contrato (pilar 5) representa la pieza de software que permitirá el control externo.

A continuación se presenta la descripción individual de las componentes más importantes:

**Servicios:** En esta componente se representan los elementos necesarios para la identificación de un servicio y clasificación de los servicios que pueden formar parte de las acciones de los contratos. Por ejemplo, nombre del servicio, identificadores, alcance, propósito, etc. Para más detalles consultar (17). El comportamiento funcional de cada servicio se expone a través de la componente Comportamiento.

**Comportamiento:** El comportamiento de un servicio se logra a partir de combinar operaciones y eventos que son representados por las componentes Operaciones y Eventos.

**Parámetros Context-Aware:** Se denomina parámetros context-aware a la representación de la información de contexto que forma parte de los parámetros de entrada de las funciones y métodos exportados por los servicios, estableciendo de esta manera una relación entre el componente Servicios y el componente Parámetros contex-aware. La influencia de estos parámetros en el comportamiento funcional de los servicios es representada a través de la relación entre los componentes Parámetros context-aware y Comportamiento.

**Contexto:** Este componente representa el contexto o información de contexto definida anteriormente en esta sección. Para el modelo planteado en esta tesis, este tipo de información es utilizada de dos maneras diferentes: 1. para la asignación de los valores que toman los Parámetros context-aware; 2. esta información puede ser utilizada para definir los invariantes que se representan en los contratos.

Otro de los elementos importantes que se tendrán en cuenta para la propuesta de solución es el tipo de representación que se determina para el entorno. Para este propósito se observará que ligado al entorno se provea una infraestructura que permita distribuir el contexto producido por las fuentes. Esta infraestructura fue denominada capa de contexto, y sirve de intermediaria entre las fuentes contextuales y los elementos que permitirán inyectar las propiedades de sensibilidad al contexto del DHD.

### 1.3. Limitaciones

Las principales limitaciones que se evidencian en el aporte de esta tesis se relacionan con la formalización de varias de las especificaciones que aquí se proponen. En este sentido, es necesario iniciar la formalización de los requerimientos de manera efectiva para captar aquellos aspectos conceptuales que determinan configuraciones particulares del DHD. Actualmente se cuenta con un recorrido en la investigación significativo en la construcción de las especificaciones del DHD, esto se desarrolla en el capítulo ?? donde se citan modelos canónicos que interpretan propiedades bien definidas para su descripción formal.

## 1.4. Organización del documento

---

En el trayecto del textual de la tesis se hace más presente que las problemáticas sobre adaptabilidad dinámica, reconfiguración e inteligencia (AdRI) están fuertemente orientadas a la construcción de reglas, lo cual implica que eventualmente pueden existir limitaciones importantes para su manipulación. En este sentido, el estudio está enfocado en cuestiones de diseño e implementación en las que se puedan lograr instancias del sistema cuya configuración resuelva cuestiones funcionales manteniendo las características de AdRI descritas en las secciones 1.1 y ??.

## 1.4. Organización del documento

**Capítulo 1:**

**Capítulo 2:**

**Capítulo 3:**

**Capítulo 4:**

**Capítulo 5:**

**Capítulo 6:**

**Capítulo 7:**

**Capítulo 8:**

**Apéndice:**

**Bibliografía:**

## 1.5. Publicaciones de Apoyo

La mayor parte de los avances de esta tesis ya fueron publicados en libros prologados, revistas con referato y memorias de conferencias y congresos, registrándose a la fecha algunas publicaciones en prensa o revisión.

Los primeros aportes, donde se comienza a desarrollar una noción sobre la inyección de contratos en una plataforma e-learning, se describe en el capítulo "Implementaciones de entornos e-learning en la formación de arquitectos. Hacia una aplicación contex-aware dinámica físico-virtual" Capítulo XIII en Rodríguez Barros, Diana. (Comp.) Experiencia Digital. Usos, prácticas y estrategias en talleres de arquitectura y diseño en entornos virtuales. Mar del Plata, Universidad de Mar del Plata, 2006, pp. 195-204.

El marco teórico y metodológico interdisciplinar que fundamenta la construcción del concepto DHD fue publicado en el libro: Hacia un Dispositivo Hipermedial

Dinámico: Educación e investigación para el campo audiovisual interactivo (61), cuya autora es la Dra. Patricia San Martín con la colaboración del autor de esta tesis conjuntamente con Griselda Guarnieri y Guillermo Rodríguez. En el capítulo 5 titulado "Sistemas Context-Aware en dispositivos hipermediales dinámicos para educación e investigación" se describe una primera aproximación de la arquitectura conceptual de un DHD con propiedades de sensibilidad del contexto. En el capítulo 6 titulado "Los contratos context-aware en aplicaciones para educación e investigación" se introduce la primera noción de los contratos sensibles al contexto y se propone un mecanismo de implementación.

Las siguientes dos publicaciones completan el recorrido sobre la conceptualización del DHD. Fueron tomadas como punto de partida para la definición de los requerimientos del DHD, propuesto en el capítulo ??

San Martín, P.; Guarnieri, G; Rodríguez, G; Bongiovani, P.; Sartorio, A. —El Dispositivo Hipermedial Dinámico: Campus Virtual UNR, publicado en el 2010 por la Universidad Nacional de Rosario. Este libro, prologado por la Dra. Graciela Carbone, plantea un proceso de reconceptualización del Campus Virtual de la Universidad Nacional de Rosario (UNR), Argentina realizado en el marco del Programa de Investigación, Desarrollo y Transferencia "Dispositivos Hipermediales Dinámicos" (CIFASIS: CONICET, UNR, UPCAM) a solicitud de la Secretaría de Tecnologías Educativas y de Gestión (UNR). El objetivo se centró en promover y fortalecer estratégicamente la integración de las TIC en actividades educativas, de investigación y vinculación tecnológica en el actual contexto físico-virtual. La metodología implementada estudió el caso conjuntamente con los propios actores de la UNR, fundamentándose en conceptos, método y bases epistemológicas de la investigación interdisciplinaria en el marco de los sistemas complejos.

San Martín, P.; Guarnieri, G.; Sartorio, A.; Rodríguez, G. —Construir un campus virtual: reflexiones sobre un caso de vinculación tecnológica. Publicado en el 2008 por la Revista de la Escuela de Ciencias de la Educación. Este artículo con referato presenta una de las primeras experiencias de implementación de la modalidad de becario e investigador en empresa ofrecida por el Consejo Nacional de Investigaciones Científicas y Técnicas -CONICET-, llevada a cabo durante los años 2006 y 2007 en una organización que si bien era de base tecnológica, deseaba consolidar su perfil como institución educativa, ya que sus servicios principales se centraban desde el año 2000, en la capacitación profesional para la operatoria de softwares multimediales, desarrollo aplicaciones hipermediales y composición hipermedial.

Otras publicaciones que colaboraron el conceptual del DHD fueron:

Guarnieri G., Rodríguez G., Sartorio A. (2007). De la máquina de Enseñar a las Interacciones Múltiples. Congreso sobre nuevas tecnologías y educación (Eduotec 2007), Buenos Aires, Argentina.

Rodríguez G., Sartorio A., Guarnieri G. (2007). El software libre en el campo del E-learning. Congreso sobre nuevas tecnologías y educación (Eduotec 2007).



## 1.5. Publicaciones de Apoyo

---

Buenos Aires. Argentina.

San Martín P., Sartorio A., Rodríguez G. (2006) Una mesa de arena para Investigar y Aprender en Contextos físicos-virtuales-interactivos-comunicacionales de Educación Superior. Actas del XV Encuentro Internacional de Educación a distancia. UDG. Guadalajara, México.

Sartorio A., Guarnieri G. (2006), Diseño de herramientas de Context Aware Dinámico aplicadas a una experiencia de taller físico-virtual-interactivo-comunicacional. Segunda edición de las Jornadas Abiertas de Informática SADIO Rosario.

En cuanto al recorrido sobre la teoría de los contratos sensibles al contexto, su implementación y metodología de aplicación, las siguientes publicaciones (en orden de importancia) sostienen lo expuesto a partir del capítulo 4.

Sartorio, A., Cristiá, M. (2009). First Approximation to DHD Design and Implementation. CLEI ELECTRONIC JOURNAL.

Rodriguez G., San Martín,P., Sartorio A. (2009), Aproximación al modelado del componente conceptual básico del Dispositivo Hipermedial Dinámico. XV Congreso Argentino de Ciencias de la Computación.

Sartorio A., San Martín P., Rodriguez G., Guarnieri G. (2009), Los contratos sensibles al contexto para los Dispositivos Hipermediales Dinámicos. Jornadas de Ciencias de la Computación - FCEIA - UNR.

Rodriguez G., Sartorio Alejandro (2008), Aspectos Tecnológicos y Modelos Conceptuales de un Dispositivo Hipermedial Dinámico. Sexto Congreso Internacional en Innovación Tecnológica Infomática Universidad Abierta Interamericana.

Sartorio A., Cristia M. (2008), Primera Aproximación al Diseño e Implementación de los DHD. XXXIV Conferencia Latinoamericana de Informática (CLEI 2008).

Todos los aportes que se brindaron a la comunidad Sakai <sup>d</sup> en la novena conferencia junto con otros significativos trabajos: <sup>e</sup>.

Sartorio A. (2008), A conceptual framework to apply Sakai with contract and its practical implementation. 9th Sakai Conference Paris, France.

Cuestiones metodológicas referidas la diseño del contrato y la detección de su ubicación en la etapa de diseño constan en los siguientes trabajos:

Sartorio, A. (2008), Un modelo comprensivo para el diseño de procesos en una Aplicación E-Learning. XIII Congreso Argentino de Ciencias de la Computación.

---

<sup>d</sup>Sakai Community:<http://sakaiproject.org/community-overview>

<sup>e</sup> <http://confluence.sakaiproject.org/display/CONF09/Conference+Presentations>

CACIC 2007.

Sartorio A. (2007), Un comprensivo modelo de diseño para la integración de procesos de aprendizaje e investigación en una Aplicación E-Learning, Congreso sobre nuevas tecnologías y educación (Edutec 2007), Buenos Aires, Argentina.

Para el capítulo 7 se utilizaron los avances sobre el uso de métricas y tipos de condicionales a partir de las siguientes publicaciones:

Sartorio, A., Rodriguez, G.(2010), Condicionales DEVS en la coordinación de contratos sensibles al contexto para los DHD. CACIC 2010. En prensa.

Rodriguez, G., Sartorio, A., San Martín, P. (2010) SEPI: una herramienta para el Seguimiento y Evaluación de Procesos Interactivos del DHD. CACIC 2010. En prensa.

Sartorio A. et al (2007), Students' interaction in an e-learning contract context-aware application with associated metric, Actas del IATED2007, International Technology, Education and Development Conference, IATED, Valencia, España.



# CAPÍTULO 2

---

## Dispositivo Hipermedial Dinámico: Marco General

---

### 2.1. Introducción<sup>2</sup>

Este capítulo presenta el marco conceptual del Dispositivo Hipermedial Dinámico (DHD) construido bajo una metodología de trabajo interdisciplinario, que ha posibilitado abordar la problemática del campo disciplinar de la presente Tesis contextualizadamente. A continuación se expone la definición conceptual del DHD:

Un Dispositivo Hipermedial Dinámico -DHD- es una red heterogénea conformada por la conjunción de tecnologías y aspectos sociales (red sociotécnica) que posibilita a los sujetos realizar con el otro acciones en interacción responsable para *investigar, aprender, dialogar, confrontar, componer, evaluar*, diseminar bajo la modalidad de taller físico-virtual, utilizando la potencialidad comunicacional, transformadora y abierta de las TIC, regulados según el caso, por una “coordinación de contratos”.

(Sán Martín, et. al., 2008)

Seguidamente se plantearán aspectos claves que han configurado la noción de DHD, observándose luego los actuales límites computacionales para dar solución a la totalidad de los Requerimientos del DHD.

## 2.1. Introducción2

---

La perspectiva teórica y metodológica del DHD se ha propuesto a través del Programa de Investigación, Desarrollo y Transferencia “Dispositivos Hipermediales Dinámicos” –<http://www.mesadearena.edu.ar>– (CIFASIS: CONICET-UNR-UPCAM), que lleva adelante desde el 2008 tareas de I+D+T. Específicamente el proyecto acreditado “obra Abierta: Dispositivos Hipermediales Dinámicos para educar e investigar” (CONICET-UNR), es el que desarrolla e implementa el estudio de la red sociotécnica en el marco de las organizaciones. Los ejes de I+D del programa son:

- La modalidad de taller físico-virtual en diversos contextos organizacionales.
- El modo interactivo-intersubjetivo de la red sociotécnica.
- La teoría de los “Sistemas Complejos” para el estudio de casos y modelado del DHD.
- La teoría de “Coordinación de contratos” para el modelado de una pieza de software bajo la perspectiva context-aware dinámico.
- Sistemas de agentes de software para actuar en ambientes dinámicos inciertos.
- Herramientas y aplicaciones Web colaborativas.
- Diseño de interfaces y heurísticas.
- Repositorios institucionales de Acceso Abierto para la publicación de materiales de Ciencia, Tecnología y Educación. Gestión de la Información.
- La formación especializada en la crítica y difusión de las artes a través del DHD.
- Participación ciudadana mediatizada por DHD

Algunos de estos ejes son transversales a los diferentes proyectos de investigación en curso del Programa y vinculan, entre otros, los principales planos que componen al DHD. Estos planos han sido diseñados para esta tesis con el propósito de facilitar la comprensión de cómo se ha elaborado el marco teórico y metodológico general. Cada uno de los planos configura elementos que permiten el análisis diferenciado desde los distintos marcos disciplinares pero que a su vez habilitan el diálogo y la construcción de lo interdisciplinar.

### 2.1.1. El plano de las interacciones del DHD

El plano 1 es abordado en la tesis doctoral “*El Modo Interactivo Del Dispositivo Hipermedial Dinámico*” cuya autora es la psicóloga Griselda Guarnieri, (Dir. Dra. Patricia San Martín, Codir. Dr. Oscar Traversa). De esta tesis se exponen a continuación los conceptos más relevantes sobre lo que se plantea como interactividad responsable en el DHD.

Según Guarnieri (2010), el problema de investigación que aborda su tesis se centra en el estudio de las interacciones que se suscitan en la trama compleja que se configura cuando los sujetos se encuentran mediados/mediatizados en su comunicación y producción por las actuales posibilidades de las Tecnologías de la Información y Comunicación (TICs).

Esto implica el estudio de las **interacciones** a través de sus definiciones, *caracterización, contextualización, medición y análisis*. Cada uno de estos elementos establece las referencias de contacto con otros hiperplanos y proveerán parte de las restricciones y requisitos que conforman los Requerimientos DHD.

### Las interacciones

De las múltiples concepciones sobre interactividad en relación a la utilización de TIC se señalarán sólo algunas que resultan significativas a las argumentaciones para una mejor comprensión de la construcción conceptual. La más elemental concibe la interactividad en el análisis de la relación entre el individuo y la computadora. Esta concepción era plausible antes del advenimiento de internet, un ejemplo de la cotidianeidad eran las situaciones lúdicas que experimentaban los usuarios en soledad con la computadora en la pasada década. Actualmente es extraño que alguien lo haga de esta forma, generalmente, las actividades de esparcimiento se realizan en red. Más allá de entrenamientos o capacitaciones específicas con sistemas expertos y de simulación con alto grado de automatizaciones que también hoy pueden estar en red, cuando se plantea la interacción bajo perspectivas constructivistas del conocimiento, se asume un sujeto dialógico constructor de contenidos, que despliega sus capacidades críticas en el propio acto responsable, acto social instituido en la presencia de un otro. Desde esta perspectiva, existen autores que han tenido en cuenta opciones más amplias para definir la interactividad y consideran que esta no se agota a la relación individuo-computadora, sino que implica también al vínculo mediado entre los sujetos.

**Definición 1** (ContratoDHD). *Un ContratoDHD es un diseño de contrato según Meyer, bajo la filosofía de "Design by Contract"*

*Retomaremos una definición de interactividad como la capacidad gradual y variable que tiene un medio de comunicación para darle a los sujetos un mayor poder tanto en la selección de contenidos (interactividad selectiva) como en las posibilidades de expresión y comunicación (interactividad comunicativa) (135).*

En este sentido, la interactividad selectiva está vinculada principalmente a las posibilidades de selección de contenidos. A diferencia de la interactividad selectiva, la comunicativa se basa en que el usuario participe de intercambios dialógicos. "En la interactividad selectiva, hay un individuo que pregunta o elige una opción y el sistema le responde automáticamente; en la interactividad comunicativa, hay un individuo emisor y otro receptor que pueden intercambiar roles. En el primer caso, el número de posibilidades que tiene el sistema de responder es -por lo menos

## 2.1. Introducción2

---

en la mayoría de los casos limitado o a veces de una única manera; mientras que en la segunda opción, la interacción es imprevisible, es decir las posibilidades de respuesta son infinitas por las características humanas de los interactuantes" (135).

Entonces, la interactividad comunicativa es mucho más difícil de cuantificar y medir que la interactividad selectiva. Sin embargo, una noción actual de interactividad debe integrar y considerar ambas problemáticas, lo cual le otorga la dimensión de un vínculo intersubjetivo mediatizado por todas las posibilidades integradas que ofrecen las TIC. En el caso de una interactividad contextualizada en procesos de educación, investigación y/o producción el vínculo intersubjetivo implica asumir dicha interactividad como acto responsable de parte de los sujetos participantes. No es sólo lo que se comunica, sino también lo que se produce tanto en forma individual como colectiva.

Es relevante observar que la interacción entre sujetos no se puede separar de la comunicación, de la reciprocidad, "del compartir". La propuesta DHD considera que ambos términos están interrelacionados, y que la interactividad integra cuestiones propias del actual contexto físico-virtual visto como red socio-técnica responsable.

En las últimas décadas, el término interactividad se ha entramado profundamente con la digitalización, tanto de información, como de contenidos, por lo tanto no se puede pensar hoy la interactividad separada de las posibilidades de intercambiar, modificar y transmitir paquetes de información on line. A esto el posicionamiento del DHD adhiere a las iniciativas mundiales de Acceso Abierto y Código Abierto a la información y conocimiento como construcción de lo público plural y democrática.

La interactividad, a su vez, se fue distanciando de los dispositivos transmisivos (unidireccionales), generando intercambios bidireccionales o multidireccionales. Son innegables las posibilidades que ofrecen las TIC, pero cabe aclarar que es necesario un profundo proceso de capacitación y resignificación de las especificidades de su implementación y esa tarea es aún más compleja dado que implica el cambio gradual de prácticas culturales que siempre son más lentas y trabajosas que la velocidad de cambio y versatilidad de desarrollo de las TIC (San Martín et al, 2010).

Siguiendo a San Matín (2003), las construcciones hipermediales y sus modos de organización son muy diversos según la epistemología del dominio en el que se inscriban. Pero su especificidad estructural está en la ausencia de un orden jerárquico que fije previamente el dominio de su lectura y en la invención de nuevas formas.

Es necesario indagar, en el caso de las tecnologías informáticas, que poseen todos los elementos para posibilitar un intercambio interactivo, si este realmente se realiza. Durante el trayecto de investigación de la tesis mencionada circunscrita al estudio del plano 1, se efectuaron análisis de diversas implementaciones en

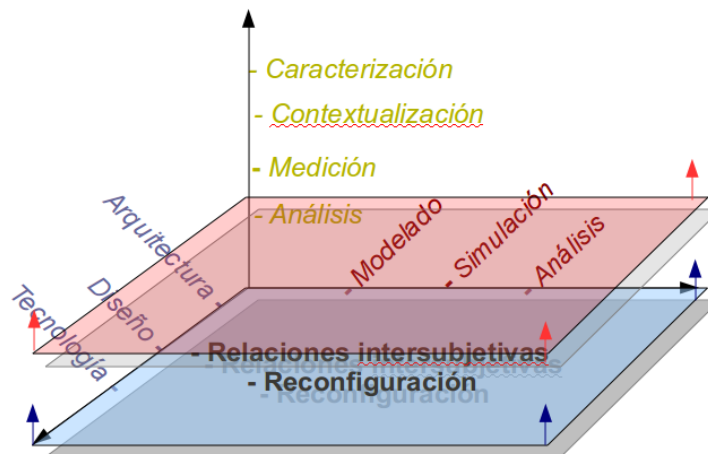


Figura 2.1: Hiperplanos del los RequerimientosDHD

el campo educativo mediatizado en distintos grados a través de internet y se ha observado frecuentemente que se reemplaza el texto impreso en papel por el digital, la fotocopidora por el texto online, pero no hay verdaderos canales que posibiliten la interactividad entre los usuarios.

**Definición 2** (Interactividad DHD). *En resumen, conceptualizamos la Interactividad DHD como un vínculo intersubjetivo responsable mediatizado por una las TIC que conforma una red sociotécnica que posibilita el intercambio y edición bidireccional o multidireccional de mensajes y objetos en un marco de trabajo colaborativo, abierto, democrático y plural.*

### 2.1.2. El plano sistémico del DHD

El plano 2 es abordado en la tesis doctoral " *La Teoría De Los Sistemas Complejos Aplicada Al Modelado Del Dispositivo Hipermedial Dinámico* cuyo autor es el Ing. Guillermo Rodriguez (Dir. Dra. P. San Martín, Codir. Dr. J.C. Gomez). Esta tesis aportó la selección de una teoría de modelado que contempla la necesidad de evaluación de las interacciones como eventos diferentes en una base de tiempo continuo. La selección del formalismo DEVS (Discrete EVent dynamics System) es presentada como una teoría específica que conjuga ambas necesidades. A su vez, se aporta el desarrollo de métricas cuali-cuantitativas en función del análisis evaluativo. La confección de la métrica fue realizada en forma interdisciplinaria y por ende se ve reflejada en varios de los proyectos de tesis en curso sobre el DHD. Entonces, del estudio sistémico del DHD se desprende que a través del *modelado*, la *simulación* y el *análisis* es posible describir aspectos importantes de su conceptualización en virtud de sus dinamismo sistémico.



### Dinamismo sistémico del DHD

Se ha planteado el concepto de interactividad directamente vinculado a las actuales posibilidades de vínculo social y herramientas tecnológicas que brindan las TIC. Las formas de trabajo colaborativo plural y democrático, la edición sincrónica y asincrónica, las varias herramientas de gestión y comunicación, la computación distribuida brindan un escenario potencialmente apto para el desarrollo de nuevas modalidades educativas, de gestión, investigación y producción.

Específicamente los sistemas hipermediales adaptativos tienen como objetivo construir un espacio capaz de ajustarse a las particularidades de cada actor, lo que constituye una forma única de interacción y reciprocidad entre el sujeto y el sistema. Su naturaleza permite configurar entornos para lograr que los participantes alcancen los objetivos establecidos mediante contenidos y recorridos adecuados a sus aptitudes, intereses y preferencias. Por lo tanto, estos sistemas buscan que el contexto se adapte al usuario y no al contrario, como sucede en los hipermediales “clásicos”, los cuales muestran el mismo contenido y los mismos enlaces a todos los usuarios. En este sentido, para que el sistema sea adaptativo debe ser configurado en un entorno hipermedial donde el usuario sea capaz de realizar dicha adaptación (153).

Existe diferencia entre estos sistemas hipermediales adaptativos y un sistema adaptable ya que éste último se enfoca, básicamente, en proporcionar al usuario herramientas para la personalización del sistema (color, tipo de letra, tamaño de letra, etc.), o en contar con interfaces para diferentes niveles (por ejemplo, experto, principiante, etc.). De este modo, la gran diferencia es que en un sistema adaptable, el usuario diseña su entorno seleccionando según sus necesidades o intereses, mientras que un sistema hipermedial adaptativo emplea un modelo de usuario para proveer adaptación automática, aunque también estos últimos pueden contar con características adaptables (153).

De modo general, se define a los sistemas hipermediales adaptativos como sistemas hipermediales que permiten personalizarse (adaptarse) en función de los usuarios individuales. De este modo, el modelo se desarrolla a partir de las metas, preferencias, características personales y conocimientos de cada usuario, el cual lo utilizará y modificará según vaya interactuando con el sistema y así adecuando la información (contenidos) y los links para la navegación, a sus necesidades, ya que considera que los usuarios aumentan y varían sus conocimientos permanentemente. Todo sistema a su vez, refiere a un contexto, por lo que cabe aclarar que por “contexto” se entiende según Dey a “cualquier información que puede usarse para caracterizar la situación de una entidad. Entendiendo a la entidad como una persona, lugar u objeto que es considerado relevante para la interacción entre un usuario y una aplicación, incluyendo al usuario mismo y la aplicación” (154).

De este modo, el contexto es información, y según como se selecciona, procesa y produce la misma a través de dichos sistemas, se posibilita la adecuación del mismo con las utilidades para el usuario. Dourish (79) sostiene al respecto que según la variable situación contextual en la cual la tecnología es utilizada, se

necesita comprender más sobre la relación entre la tecnología informática y el contexto en la que se encuentra.

El problema del contexto no refiere a cómo se representa sino que tiene que ver con un problema de relaciones, es decir, con qué se vincula la actividad que se desarrolla, de las características del contexto en que se desenvuelve y a su vez, cómo y porqué las personas que interactúan pueden tener una comprensión mutua del contexto para sus acciones. La idea de contexto consiste en un conjunto de características del mismo que rodean las actividades que se realizan, y que estas características pueden ser codificadas y hacer viable, de este modo, un sistema de software. Entonces, el contexto es lo que las personas hacen y no la sola descripción de un escenario. Es un producto de la relación, mas que una premisa. Así el contexto y la actividad no pueden separarse. El contexto no es algo estable, dado, como una descripción externa a la actividad misma. Por el contrario, aparece y es sostenido por la misma actividad.

En síntesis, en el marco de este trabajo, el DHD debería ser sensible o adaptable al contexto para posibilitar el dinamismo propio de las interacciones dadas en las comunidades de práctica.

### 2.1.3. El plano de la Ingeniería Computacional del DHD

El plano 3 (figura 5.3) conforma la base tecnológica y computacional del DHD, establecida por la arquitectura, el diseño y la tecnología utilizada para el desarrollo. Estos tres elementos involucran las resoluciones funcionales de los requerimientos DHD.

De esta manera se establece que en el marco tecnológico y disciplinar de los siguientes capítulos que componen la presente tesis, cuando se refiere al Dispositivo Hipermedial Dinámico se limita el alcance solamente a los requerimientos funcionales, computacionales y de diseño, dando cuenta de la imposibilidad de brindar soluciones a la totalidad de los elementos que devienen de la construcción interdisciplinaria (2.1).

### Campo teórico-metodológico interdisciplinario del DHD

Las tareas de investigación llevadas adelante desde el 2003 por el grupo de investigación Obra Abierta en sus diferentes estudios de caso, refieren al estudio de problemáticas vinculadas a la integración efectiva de las TIC, en diferentes contextos organizacionales educativos tanto públicos como empresariales (San Martín et al, 2008) fundamentándose en conceptos, método y bases epistemológicas de la investigación interdisciplinaria en el marco de los sistemas complejos (García, 2007).

Estos fundamentos posibilitaron, en la continuidad de los distintos proyectos, la construcción de la noción de Dispositivo Hipermedial Dinámico (DHD) y la conformación del programa ya mencionado que tiene por objetivo general consolidar el marco teórico, metodológico y de desarrollo tecnológico del DHD. En este sentido, el abordaje de las problemáticas se realiza como se expuso en el parrafo anterior, implementando una metodología de investigación interdisciplinaria que estudia el mismo como sistema complejo.

Siguiendo a San Martín y Guarnieri (2009), construir una red sociotécnica en un contexto físico-virtual con “presencialidad” subjetiva es sentir la Presencia del otro que cobra sentido en el compromiso de participación responsable para enseñar y aprender, investigar y/o producir cualquiera sea el grado y tecnología de mediatización. Presencia que en su dimensión simbólica posibilita el vínculo intersubjetivo y da lugar a la distancia necesaria para la interrogación, la representación como “juego al pensamiento” (Enaudeau, 1999), la mediación de los discursos (Lasch, 2005) y la acción (Bruner, 2001). La utilización o no de TIC en un determinado proceso educativo, investigativo o de producción y el grado de mediación/mediatización de dicho proceso no puede ignorar el contexto del siglo XXI, que da muestra de los múltiples impactos de la globalización. El problema relevante en una sociedad donde la interactividad digital es un hecho consumado, es interrogar sobre cómo sostener la presencia subjetiva y la participación ciudadana responsable e inclusiva. Estas dimensiones, pensadas como indisociables, podrían posibilitar quizás la reflexión sobre la construcción de nuevos vínculos generadores de “civitas” (Borja, 2006) más allá del grado de mediación/mediatización. Entonces, la problemática así planteada excede el tradicional dualismo presencial - a distancia y lo meramente cuantitativo del grado de mediatización, centrándose el debate en su real dimensión política dada la emergencia de nuevas formas culturales físico-virtuales de gestión, transmisión, producción y acceso -o no- a la información y conocimiento. Esto involucra de hecho las posibilidades de desarrollo tecnológico colaborativo y la extensiva utilización abierta de herramientas digitales aptas para múltiples propósitos.

Partiendo de estos posicionamientos, se elaboraron aspectos referidos a las condiciones necesarias para el que hacer interdisciplinario en el propio campo de acción, atendiendo a los fundamentos que sostienen la construcción del marco conceptual común y a reflexiones acerca del desarrollo de una práctica convergente que supone asumir una cierta distancia hacia los problemas particulares de los propios campos disciplinares para lograr visualizarlos desde otras perspectivas menos conocidas situadas en el contexto sistémico que se expresan. Este quehacer interdisciplinario demanda un necesario corrimiento generador de tensiones tanto a nivel subjetivo como intersubjetivo entre la propia formación profesional especializada y la tarea interdisciplinaria que habita en un caso que muestra una realidad dinámica y dificultosa que no se detiene, constituyéndose dichas tensiones durante el desarrollo de la investigación en un aspecto revelador de la integración dinámica del grupo de trabajo. Siguiendo a García (2007) se acuerda que la forma en que se van generando esas necesarias tensiones puede condicionar significativamente la calidad de los resultados producidos interdisciplinariamente. Surgen así interrogaciones sobre las posibilidades y limitaciones de ese necesario corrimiento y se desarrolla la construcción de una metáfora del DHD: “la mesa de

arena". Esta metáfora ha sido presentada por San Martín y Guarnieri (2009) como plano de pensamiento del arte, proyectándose como interferencia intrínseca al plano filosófico del concepto de dispositivo y al plano científico de los sistemas complejos y, como interferencia extrínseca del pensamiento creativo tecnológico. En referencia directa a esta tesis, esto significa que las cuestiones tecnológicas y computacionales están fuertemente involucradas en los requerimientos de los DHD, pero son insuficientes para el abordaje del plano filosófico sobre lo que se entiende conceptualmente como dispositivo.

La noción de dispositivo, como señala Traversa (2001), tiene un campo de aplicación extenso y su empleo puede encontrarse en disciplinas tales como la filosofía, la mecánica, la informática, la comunicación, la sociología, la educación, el arte, etcétera, presentando perspectivas analíticas diversas (Deleuze, 1990; Agamben, 2007; Meunier, 2007). Sus alcances abarcan desde mecanismos y componentes tangibles a configuraciones de un alto grado de abstracción. En este último sentido Foucault lo conceptualiza como un conjunto heterogéneo:

"(...) lo que quisiera señalar en el dispositivo es justamente la naturaleza del vínculo que puede existir entre esos elementos heterogéneos. (...) entre dichos elementos –discursivos o no discursivos- existe algo así como un juego, cambios de posición, modificaciones de funciones, que pueden, también ellos, ser muy diferentes". (Foucault: 1991, 185).

Siguiendo a Foucault, se adopta una visión que asume lo heterogéneo de los elementos y sus modos de vinculación que entran en concepciones de poder y sujeto del saber. El dispositivo se manifiesta entonces, como una entidad compleja compuesta por la integración de dos dimensiones indisociables: una técnica (o conjunto de técnicas constructivas que comportan una materialidad y una configuración particular) y una social dada por las relaciones intersubjetivas y la situación en la que se inscriben. Este concepto da cuenta de la presencia dinámica de tecnologías, vínculos interactivos/intersubjetivos y representaciones como lugar en que operan los intercambios discursivos, potencial espacio para la co-enunciación ya que se torna posible el emplazamiento social de los discursos.

Retomando la metáfora de "la mesa de arena" como lugar de interacción múltiple, léase un plano sustentador físico-virtual que posibilita realizar responsablemente ensamblajes de elementos heterogéneos a través de operaciones complejas, donde es posible conversar y dialogar mientras se construye e integra conocimiento con ideas creativas y diversos elementos, materiales o herramientas disponibles; el DHD propone habitar un lugar de acción reflexiva, de enunciado de hipótesis, de observación detenida, de participación activa para vivenciar la virtualidad de las ideas y a la vez admitir el error para volver a comenzar. Así, se hace imprescindible vincular la dimensión contextual de cada sujeto participante en su más amplio sentido, como variable que condiciona los procesos y calidad de las interacciones. Se trata entonces, de habilitar la evaluación reflexiva y continua sobre lo que se aprende, se enseña, se investiga y se produce: el valor y el sentido de lo que se comunica y disemina para construir "civitas". De allí, este enfoque difiere cualitativamente de la administración, gestión y comunicación de la información, dado

## 2.1. Introducción2

---

que si bien la incluye, atiende a una dimensión ética que excede lo meramente informacional.

En la metodología de trabajo propuesta, los conceptos guían a los procesos y a la interevaluación permanente de los mismos y de sus productos asociados como dinámica propia del desarrollo de conocimiento y del compromiso hacia el otro. Así, el despliegue de la red sociotécnica, trasciende lo meramente instrumental, posibilitando mediaciones y mediatizaciones sustentadas en un conocimiento profundo de los propósitos que habilitan la construcción de un campo de responsabilidad efecto de la interacción social participativa, en el marco del constructivismo dialéctico (Vygotsky; 1988) y la modalidad pedagógica de taller. En este sentido San Martín (2009), ha desplegado una visión no instrumental y crítica considerando a las “competencias profesionales” requeridas, como capacidades que se sintetizan en el sujeto como el “saber hacer – saber ser”, que más allá de la singularidad disciplinar se construyen e integran en profundidad como un saber “ha-ser” ético, manifestado en una actitud responsable hacia la calidad de la existencia asumiendo la diversidad y la complejidad del contexto físico-virtual, expresándose en los planos científicos, artísticos y técnicos a través de articuladas coordenadas de acción. La noción de DHD para educar, investigar y producir en el contexto físico-virtual conceptualiza a la educación y/o investigación (cualquiera sea su grado de mediación/mediatización) como proceso complejo que involucra la constitución misma de los sujetos que la piensan, en su más profunda dimensión ética.

Entonces, para favorecer la alternativa del saber “ha-ser” ético, se hizo necesario construir una perspectiva organizacional, pedagógica, investigativa y de desarrollo tecnológico, poniendo en obra una metodología interdisciplinaria que sustentara el estudio del sistema complejo para abordar los problemas desde su contexto real situacional. Esta aproximación solicita teorías y técnicas reflexivas de acción participativa por parte de todos los actores involucrados para el tratamiento de los mismos en su complejidad. El supuesto se basa en que dicha participación físico-virtual en el propio tratamiento metodológico de las problemáticas sustentada en el marco epistemológico de los sistemas complejos, habilitaría la necesidad de desarrollo de originales síntesis de acción (multiplicidad metodológica-prácticas convergentes) posibilitando tal vez la alternativa de enfrentarse y afrontar la responsabilidad ética, asumir responsabilidad por esta responsabilidad.

Cabe aclarar que esta perspectiva sobre los sistemas complejos, confronta con aquellas “soluciones” informáticas donde sistemas tutoriales inteligentes (STI), determinan pasos claves en la metodología de una propuesta educativa, investigativa y/u organizacional centrada en la linealidad causa-efecto. Es evidente también que no se adhiere a soluciones donde el diseño “instruccional”, desarrollo de contenidos, materiales didácticos (objetos digitales educativos) y administración de plataforma, están bajo un modelo terciarizado quedando determinaciones y desarrollos claves a cargo de personas ajenas a la institución/organización. En ambos casos, es observable que no se capitalizan las posibilidades de desempeño profesional y crecimiento de los responsables (la trama institucional conformada por todos sus actores, especialmente docentes, investigadores y profesionales), al reconocimiento de la diversidad y necesario vínculo intersubjetivo singular con quienes

se están formando, evidenciándose a nivel institucional, la ausencia del diseño y desarrollo de una política académica de formación permanente, de promoción del diálogo y reconocimiento de las propias capacidades de la organización.

Dado todo lo expuesto, en la observación del plano 2 de la figura 5.3 se visualiza que:

“La complejidad de un sistema no está solamente determinada por la heterogeneidad de los elementos (subsistemas) que lo componen, y cuya naturaleza los sitúa normalmente dentro del dominio de diversas ramas de la ciencia y la tecnología. Además de la heterogeneidad, la característica determinante de un sistema complejo es la interdefinibilidad y mutua dependencia de las funciones que cumplen dichos elementos dentro del sistema total. Esta característica excluye la posibilidad de obtener un análisis de un sistema complejo por la simple adición de estudios sectoriales correspondientes a cada uno de los elementos.”

(García, 2007,87)

Por lo tanto, como exponen Rodríguez y San Martín (2010), la funcionalidad global de un sistema se da precisamente por las interacciones y no se encontrará tal funcionalidad si se observan solamente algunos elementos. Es debido a esto que dicha funcionalidad se denomina “emergente”, dado que solo se encuentra a nivel sistema (Bar-Yam, 1997). Además, el sistema como totalidad es abierto, no tiene contornos rígidos; está inmerso en una realidad más amplia con la cual interactúa por medio de flujos heterogéneos de categorías diversas.

El sistema complejo no es algo ya dado que no hay más que observar y analizar, sino que demanda un esfuerzo en la investigación para su conceptualización como recorte posible de una realidad mucho más amplia e indefinible en sus límites. Esta construcción demanda una metodología que de lugar a la formalización de modelos sucesivos donde se busca una aproximación que sea suficientemente coherente en la capacidad de explicar el funcionamiento de dicha construcción (sistema complejo) dando cuenta de los hechos observados. Esta posibilidad de modelización del sistema complejo permite comprender con mayor exactitud como el pensamiento científico conforma un plano de referencia mediante proposiciones. Modelizaciones que se definen a través de expresiones matemáticas, formalismos u órdenes metodológicos híbridos cualitativos y cuantitativos donde el grado de variabilidad para definir la complejidad no corresponde a lo inconmesurable ya que lo complejo no es científicamente análogo a lo caótico, aunque perceptualmente lo pareciera.

La multiplicidad de miradas analíticas y disciplinares que habilita el DHD, permite observar que en el marco teórico de los sistemas complejos refiere a un estado de cosas en un tiempo finito y en determinada realidad contextual referenciable donde se puede accionar grupalmente para efectuar cambios estructurales constatables, mientras que a su vez en un plano absolutamente diferenciado, el DHD

## 2.2. Requerimientos DHD

---

resuena como acontecimiento en el pensamiento y subjetividad de los participantes. Así surge el desafío de articular el intercambio con el otro (disciplinar) para descubrir, explorar y construir grupalmente alternativas metodológicas que desde cada singularidad desplieguen posibles “interferencias” intrínsecas o extrínsecas para elaborar lo “significativamente común”.

Como dinámica de intercambio colaborativo, la modalidad de taller físico-virtual permite un ámbito de aplicación donde el conocimiento se construye participativamente tensionando con otros y con uno mismo en la propia interrogación. Se plantea esta modalidad de trabajo como imprescindible para la construcción de un contexto social y productivo físico-virtual inclusivo y democrático propio de lo público. Atendiendo al plano de pensamiento de desarrollo tecnológico, es posible desde lo disciplinar visualizar las herramientas y sistemas informáticos que posibilitan trabajar colaborativamente en el actual contexto físico-virtual. Lo propuesto, intenta aportar un posible camino hacia el análisis evaluativo sobre cómo se desarrollan y se podrían mejorar procesos de interactividad responsable a través de redes sociotécnicas para educar, investigar y producir.

## 2.2. Requerimientos DHD

Esta sección se referencia en el marco de los procesos de la Teoría de Requerimientos (AGREGAR CITA) para abordar la problemática conceptual DHD que presenta aspectos diferenciales a los usuales de requerimientos TIC.

Se adopta una diferenciación sutil entre los términos “Requisitos” y “Requerimientos” que expresan dos niveles relacionados cuando se abordan las resoluciones funcionales, relacionales y reconfigurativas del DHD. Este planteo se realiza siguiendo a diferentes proyectos (155; 156; 157; 158) que utilizan en forma similar esta distinción. En este sentido se incluyen ítems que tratan cuestiones significativas:

1. Determinación de Requisitos/requerimientos que total o parcialmente no tienen solución computacional.
  - a) Aspectos de reconfiguración
  - b) Aspectos de interactividad responsable
2. Determinación del propósito que impide la solución computacional.

**Definición 3** (Requisito DHD). : *Entonces, definimos como requisitos del DHD (RequisitosDHD) a todas las características que implican tanto los aspectos sociales como tecnológicos del DHD. Esto significa que si se representa una secuencia de “ejecución”, por ejemplo situada en un proceso educativo (59), a través de una máquina de estado, la representación de cada uno de los estados deberían ser descriptos a partir de los conceptos propios del DHD referidos a interactividad*



*responsable, modalidad taller, dinamismo, reconfiguración, coordinación, contextualización, etc.*

Con la finalidad de ejemplificar Requisitos del DHD se plantea como escenario un curso de formación docente destinado a la elaboración de materiales educativos digitales, donde los destinatarios deben realizar reediciones colaborativas sobre Objetos Digitales Educativos (ODE) disponibles en Acceso Abierto.

■ **Ejemplo 1.** *RequisitosDHD 1: El proceso físico-virtual de recontextualización y reedición colaborativa que realizan los alumnos sobre un ODE dado, debe quedar registrado para posibilitar la evaluación del grado de interactividad responsable de los alumnos involucrados. En el desarrollo de dicho proceso, si es necesario, el/la docente debe tener la posibilidad de cambiar dinámicamente estrategias didácticas, actividades relacionadas y la semántica de las interfases sin que estas últimas sufran modificaciones.*

**Definición 4** (Requerimiento DHD). : *Definimos como requerimientos del DHD a la interpretaciones de los RequisitosDHD que puedan ser expresadas mediante modelos formados por componentes, tipos de componentes, relaciones y tipos de relaciones que sirvan para especificar funcionalidades, conceptos, patrones y estilos. Por cada RequisitoDHD puede haber varios RequerimientosDHD.*

Como RequerimientosDHD derivados de los RequisitosDHD del ejemplo anterior se mencionan:

■ **Ejemplo 2.** *RequerimientosDHD 1: El sistema debe proveer el mayor grado de información posible sobre las interacciones de la red sociotécnica, el porcentaje cuanti-cualitativo en el uso de los servicios de edición de las herramientas seleccionadas en el entorno virtual y el porcentaje de interacciones responsables que promovió cada usuario. A su vez, se debe posibilitar que el usuario docente pueda cambiar en tiempo de ejecución la funcionalidad de los servicios del foro a partir de información de contexto del usuario alumno.*

### 2.2.1. Breves antecedentes sobre el tratamiento de requerimientos 2

El tratamiento de requisitos/requerimientos es el proceso mediante el cual se especifican y validan los servicios que debe proporcionar el sistema así como las restricciones sobre las que se deberá operar. Consiste en un proceso iterativo y cooperativo de análisis del problema, documentando los resultados en una variedad de formatos y probando la exactitud del conocimiento adquirido (Ferreira y Loucopoulos, 2001). La importancia de esta fase esencial puesto que los errores



## 2.2. Requerimientos DHD

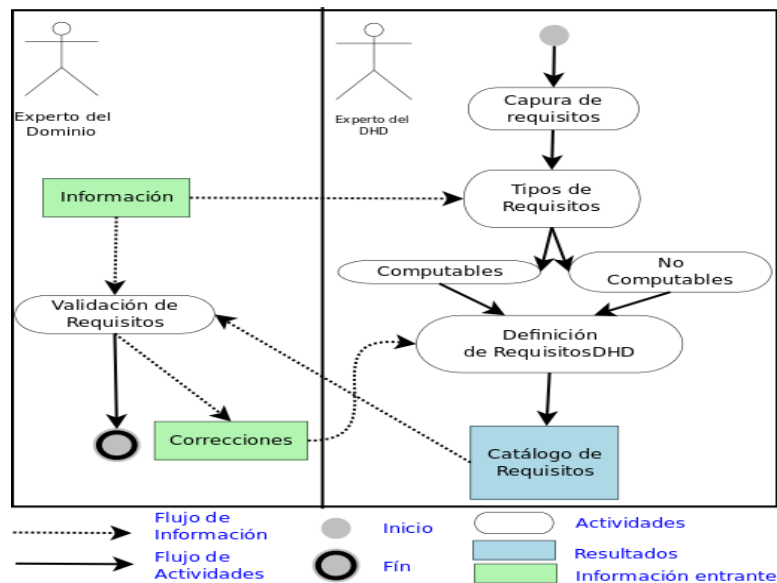


Figura 2.2: El proceso de Ingeniería de Requisitos

más comunes y más costosos de reparar, así como los que más tiempo consumen se deben a una inadecuada ingeniería de requisitos.

Dados los límites de esta tesis, no se pretende desarrollar la problemática de la Teoría de Requerimientos en general, sino sólo la breve descripción de algunos de los procesos que se adecúan mejor a los requerimientosDHD. Este abordaje permitirá exponer una caracterización indirecta de los requerimientosDHD, proporcionando un aporte enriquecedor sobre otros requerimientos convencionales inherentes a la construcción de herramientas educativas basadas en tecnología Web. Según Lowe Hall (1999), el proceso de especificación de requisitos se puede dividir en tres grandes actividades:

1. Captura de requisitos
2. Definición de requisitos
3. Validación de requisitos

### Captura de requisitos para los DHD

La captura de requisitos es la actividad mediante la cual el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema (Díez, 2001). En los DHD el proceso de captura de requisitos puede resultar complejo, principalmente por el marco conceptual complejo que sustenta la red sociotécnica para educar, investigar y/o

producir. Es observable la gestión de procesos dinámicos que requieren reconfiguraciones propias de la modalidad participativa y colaborativa de taller. Esto supone tanto a nivel tecnológico informático proporcionar un sistema flexible como a nivel organizacional diseñar e implementar políticas de participación responsable que implican altos niveles de abstracción no computables.

A continuación se exponen sintéticamente dos técnicas para la captura de requisitos que resultan pertinentes al DHD, a saber:

Aunque inicialmente se desarrolló como técnica para la definición de requisitos (Jacobson, 1995), algunos autores proponen casos de uso como técnica para la captura de requisitos (Pan, Zhu Johnson, 2001 y Liu Yu, 2000). En ingeniería del software, un caso de uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. Normalmente, en los casos de usos se evita el empleo de jergas técnicas, prefiriendo en su lugar un lenguaje más cercano al usuario final. En ocasiones, se utiliza a usuarios sin experiencia junto a los analistas para el desarrollo de casos de uso. Es en este sentido que la metodología adoptada en varios de los proyectos que integran el Programa de I+D+T "Dispositivos Hipermediales Dinámicos" considera centralmente el estudio de caso contextualizado.

Siguiendo a Thomas et al (2008), en la perspectiva DHD se considera que los procesos de producción y construcción social de la utilidad y el funcionamiento de las tecnologías son indisociables y se configuran a partir de relevantes intervenciones y estilos locales, tanto en el plano de la innovación tecnológica como del desarrollo cognitivo ya que la utilidad de un artefacto o conocimiento tecnológico está presente tanto en el diseño del artefacto como en los procesos de re-significación de las tecnologías en los que participan diferentes grupos sociales relevantes (investigadores, tecnólogos, usuarios, funcionarios públicos, integrantes de ONG, etc.).

Esta técnica fue especialmente creada para la toma de requisitos de un DHD con finalidad educativa. Está orientada hacia la creación de las reglas de coordinación de los contratos y a la identificación del lugar donde debe conectarse el contrato teniendo en cuenta el flujo de utilización de componentes dentro de un proceso educativo. En (59) se presenta un primer diseño compresivo para el modelado de los procesos de educación e-learning (Pe-Irn) en un Aplicación Web E-learning con la inclusión de contratos con propiedades context-aware (57). El modelo está basado en UWAT+ (Distante, 2004), una versión extendida y adaptada de "UWA Transaction Design Model" para el diseño de transacciones en aplicaciones Web. El principal aporte de UWATc+ es el acercamiento de un modelo útil para la representación de transacciones e-learning en una AWe-Irn; permitiendo una mejor distinción de la ubicación (entre servicio-usuario(s) y servicioservicio(s)) de la componente contrato dentro del flujo de ejecución. Detalles de este desarrollo serán expuestos en el capítulo ....XXX

### 2.2.2. Abstracción para el refinamiento de los RequerimientosDHD

Por último es necesario hacer un refinamiento de los requerimientos anteriormente mencionados para alcanzar una especificación funcional. De esta manera, se pretende exponer cuáles son los elementos de primera clase que integran las especificaciones y que son productos finales de refinamientos de otros elementos que a su vez derivan de RequisitosDHD y RequerimientosDHD.

En la figura 2.2 se muestra una primera aproximación de dichos elementos que intervienen en la etapa de refinamiento de los requerimientos hacia la especificación funcional. Cada uno de los bloques representan diferentes niveles de refinamiento, posibles de asociar a niveles de abstracción. El proceso de refinamiento de los elementos se realiza sobre los componentes abstractos del *Nivel 2*, a los cuales se les aplica unas primitivas de refinamiento para derivar los respectivos modelos concretos. Dos tipos de refinamiento distinguen este proceso:

se realiza sobre el modelo abstracto de roles cualificado con unas relaciones de transformación que permiten obtener el modelo concreto de roles. Se utilizan primitivas de refinamiento como agregación, especialización, relación, que determinan el tipo de transformación que se aplica sobre los objetos que intervienen con sus instancias en la formación estática que fuerzan las relaciones (2.1.1) para derivar los objetos de diseño.

se realiza sobre el modelo abstracto de colaboraciones con el propósito de transformar los servicios definidos sobre los objetos que intervienen en las relaciones (2.1.1). Un servicio que en un objeto de negocio es visto como un evento de ocurrencia instantánea, se transforma en un conjunto de eventos sobre los objetos de diseño, que en DHD recibe el nombre de objetos del proceso interactivo DHD (OPIDHD) 2.2.2.

**Definición 5 (OPIDHD).** *Se define a los OPIDHD como el conjunto de objetos donde sus instancias en relación permiten resolver tareas relacionadas lógicamente llevadas a cabo para lograr una actividad visible para algún sujeto (eventualmente puede ser un usuario final o Stateholder) en el DHD (ActividadDHD). Cada ActividadDHD bajo la perspectiva de un sujeto DHD tiene sus entradas, funciones y salidas.*

En el nivel 1 se encuentran las componentes tecnológicas que tienen una influencia directa en la conformación de las componentes del nivel 2. En este caso se representa con una flecha discontinua la Interactividad DHD (*Interactividad DHD*) (2.1.1) tienen una dependencia de concretización tecnológica desde los objetos que caracterizan a los sujetos (*Sujetos*) del DHD. De la misma manera la reconfiguración que se producen en los DHD depende de la posibilidad de representación y acciones sobre el contexto. A su vez, todas las funcionalidades computables son producto de la manipulación que se puedan lograr a partir de las implementaciones de los servicios.

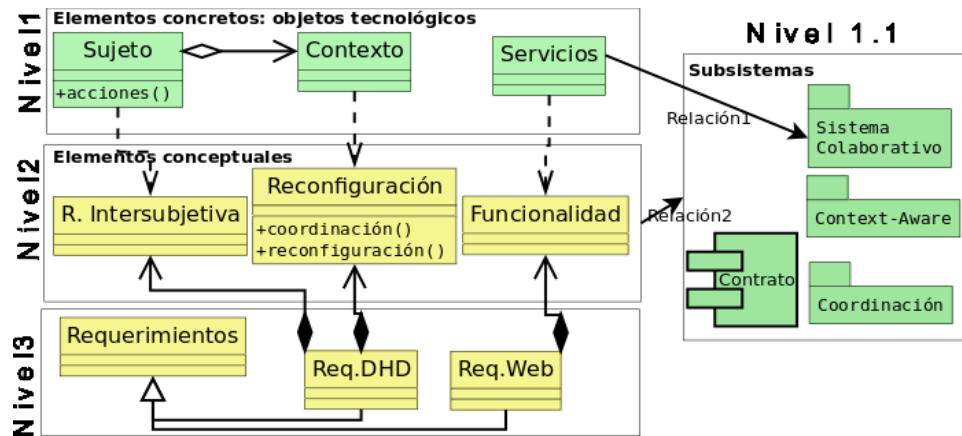


Figura 2.3: Niveles de abstracción y elementos de refinamiento sobre los RequerimientosDHD

Con el propósito de reforzar la definición de requerimientos para los DHD, el nivel 3 se utiliza para distinguir cuales son los elementos concretos (del nivel 2) en los cuales se basan los límites impuestos al concepto del Dispositivo Hipermedial Dinámico para hacer posible las representaciones computacionales.

Por otro lado, en el nivel 1.1 se encuentran los elementos que se agregan a los del nivel 1 para aportar mejor representación, diseño e implementación de los RequerimientosDHD. Se plantea que las formas de relacionamiento de estos elementos con los anteriores establecen un modelo que fija el diseño computacional del DHD justificando los argumentos expuestos en la definición del plano 3 (2.1.3). Estructuralmente, dicho modelo respeta la lógica de que indefectiblemente debe existir un nuevo nivel de representación de las funcionalidades, requerimientos y requisitos en la construcción del DHD visto desde el aporte multidisciplinario.

La figura 2.2 muestra tres subsistemas relacionados dentro del nivel 1.1 y una componente (*Contrato*) que tendrá un protagonismo conceptual importante en esta tesis. Las cuestiones de diseño, implementación y uso de los subsistemas serán tratadas en los capítulos siguientes. Se desarrollará cómo a partir de un framework colaborativo Web son utilizados sus servicios bases (relación 1). A su vez, los elementos de reconfiguración estarán ligados a un mecanismo de coordinación a partir de la intervención de los contratos y se tendrá otro subsistema encargado de la manipulación y representación de contexto. Estos mismos elementos intervendrán en el tratamiento de la Interactividad DHD donde su nivel de expresión quedará supeditado a la forma de relacionarlos.

Finalmente hemos arribado a una representación sobre la estructura que determina a los Requerimientos del DHD donde se expresa la necesidad de considerar permanentemente lo no computable para dar sentido a lo computable y efectuar las distinciones de Requerimientos tecnológicos correspondientes a través de sucesivos refinamientos.

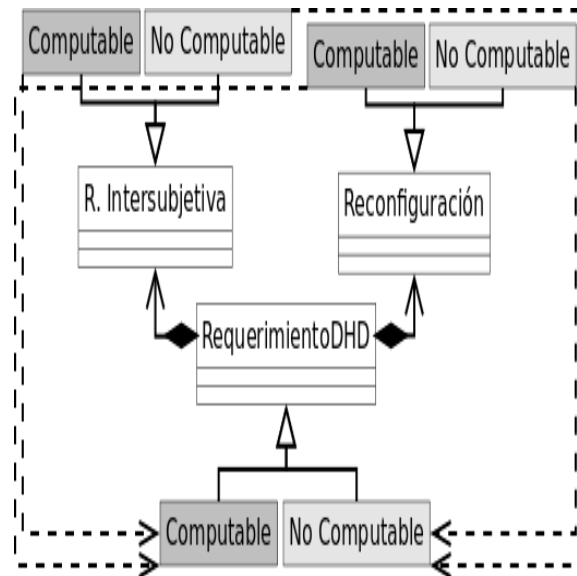


Figura 2.4: Estructura del RequerimientoDHD

## 2.3. Conclusiones

En este capítulo se hizo necesario exponer integralmente el marco conceptual del DHD para educar, investigar y/o producir desarrollado bajo una metodología de trabajo interdisciplinar con la finalidad de situar los límites disciplinares de la presente Tesis. A partir de una observación centrada en lo que puede ser computable o no, fue posible aplicar la Teoría de Requerimientos para avanzar metodológicamente en el abordaje del tratamiento tecnológico. Entonces, se logró identificar un patrón que expresa la estructura de los RequerimientosDHD. Este avance posibilita y postula con claridad el recorte disciplinar necesario al concepto general del DHD, para comprender el desarrollo de los siguientes capítulos que configuran este escrito.

# CAPÍTULO 3

---

## Estado del Arte

---

### 3.1. Introducción

Este capítulo está dedicado a la exposición de los sistemas y estructuras que fueron utilizados para la construcción de una propuesta de solución que determina un modelo evolutivo. Se presenta entonces, lo significativo del trayecto realizado en el estudio del estado del arte, partiendo de la transformación de *información de contexto* como componente de primera clase y visible en una arquitectura 4, tratado a través de un sistema especializado para su uso.

De la misma manera, se pone a otro elemento de primera clase con su correspondiente sistema que permite su manejo brindando conexión con los anteriores. Así, esta combinación de elementos y sistemas configuran una estructura de evolución que sustenta parte de la arquitectura del DHD.

### 3.2. El contexto en el DHD

En el capítulo 1 se referenció al contexto en virtud de las propiedades de sistemas especializados en usar información de contexto para brindar funcionalidades. Además, el contexto aparece como uno de los elementos, junto a los parámetros context-aware, que forma la pirámide de componentes de la solución propuesta (figura 1.2). Luego, en el capítulo 2 el contexto forma parte de la definición conceptual del DHD como un factor a tener en cuenta para la adaptación debido a

## 3.2. El contexto en el DHD

---

su influencia en los RequerimientosDHD (véase ejemplos).

En el presente capítulo, se abordará la caracterización del contexto teniendo en cuenta la taxonomía diseñada en el DHD. Esta forma de representación posibilita describir al contexto a través de una interpretación necesaria para su tratamiento tecnológico.

### 3.2.1. Caracterización del Contexto

En esta tesis se referencia al contexto para cada una de las representaciones considerando tanto los diferentes diseños para modelarlos según su tipo dependiendo de las necesidades, como su transformación en información (información de contexto) para las interfaces de los mecanismos de los sistemas y/o subsistemas sensibles al contexto.

#### Origen del contexto en el DHD

En la figura 3.1 se representan los orígenes de la conformación del contexto en el DHD. Las componentes grises son precisamente las diferentes fuentes donde se obtiene información de contexto. Por otro lado, las demás componentes de color blanco tienen una influencia en los procesos de su construcción.

Existen diferentes categorías de contexto en relación a características de tipología dinámica o estática. En este sentido, los *roles* y *permisos* asociados a los diferentes tipos de usuarios (*Miembros en formación* y *Grupo Responsables*) ayudan a caracterizar parte del perfil de un usuario. Esto permitirá ajustar determinados tipos de acciones para las diferentes herramientas (wiki, foro, videoconferencias, etc). Los servicios (ej.: agregar, borrar, etc.) que se consideran partes de las herramientas, deberán ser sensibles a la influencia del entorno teniendo en cuenta la información sintetizada que de este deviene. En este caso, en el entorno del DHD está caracterizada por información que se puede representar a través de los siguientes elementos:

- WebServer: información que se obtiene a partir de los *logs* del servidor Web que permite la ejecución de la aplicación web colaborativas del DHD. También es utilizada información de la configuración sobre el modo de ejecución, puertos habilitados, librerías, componentes, aplicaciones instaladas, políticas de seguridad, tipos de sesiones, manejo y notificación de errores, tipos de conexiones, etc.
- Objetivos: información relacionada a los objetivos que se persiguen a partir de ciertas configuraciones de los ambientes colaborativos del DHD. Estos objetivos deben estar relacionados con los RequisitosDHD y/o Requerimien-

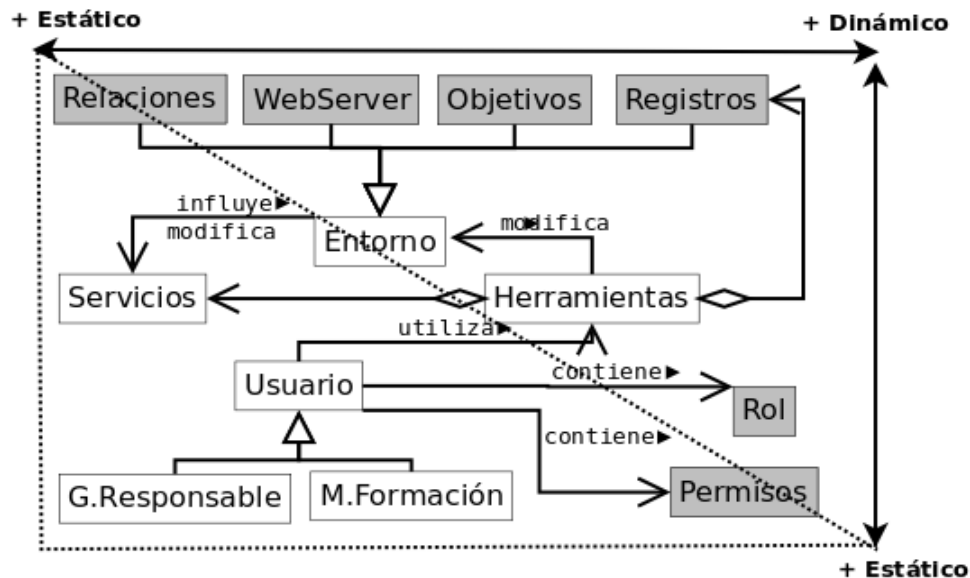


Figura 3.1: Fuentes del contexto del DHD

tosDHD en los que se encuentran involucrados Stakeholder <sup>a</sup> de forma directa o indirecta.












- Registros: información generada a partir de las diferentes intervenciones de los usuarios con las herramientas que componen un espacio colaborativo <sup>b</sup>. Habitualmente esta información aparece en forma de tablas donde se almacena, entre otras, identificador de usuario, tiempo, la herramienta y el servicio utilizado.
- Relaciones: información que permite interpretar interactividades DHD a través de la descripción de eventos secuenciados que indican relaciones entre usuarios, herramientas, servicios, información de contexto, sitios y otros tipo de información tecnológica comunicacional.
- Rol: información asociada con los usuarios de los sitios que definen diferentes perfiles de usuarios para establecer el rol que cumplen en los procesos colaborativos. Como ejemplo de los tipos de roles se pueden mencionar a los “Miembros en formación”, “Grupos Responsables”, “Administradores”, etc (57).
- Permisos: se relaciona al tipo de accesibilidad que un usuario tiene en las herramientas de los distintos espacios colaborativos. Particularmente en Sakai los permisos están directamente asociados con los roles. En la figura ?? se muestra un ejemplo tomado de la propia interfaz Sakai donde se visualizan

<sup>a</sup>Stakeholder es un término inglés utilizado por primera vez por R. E. Freeman en su obra: “Strategic Management: A Stakeholder Approach”, (Pitman, 1984) para referirse a «quienes pueden afectar o son afectados por las actividades de una empresa».

<sup>b</sup>El espacio colaborativo (Collaborative Learning) es un conjunto de métodos de instrucción y entrenamiento apoyados con tecnología así como estrategias para propiciar el desarrollo de habilidades mixtas (aprendizaje y desarrollo personal y social) donde cada miembro del grupo es responsable tanto de su aprendizaje como del de sus compañeros.



## 3.2. El contexto en el DHD

Nombre 	Agregado	ID	Créditos	Rol	Estado	Borrar
Administrator, Sakai ( admin )				Instructor 	Activo 	<input type="checkbox"/>
Canales, Roberto ( robertocanales )				Student 	Activo 	<input type="checkbox"/>
Hernández, Daniel ( danielhernandez )				Teaching Assistant 	Activo 	<input type="checkbox"/>
Pérez, Alejandro ( alejandroperez )				Instructor 	Activo 	<input type="checkbox"/>
Pérez, Pepito ( pperez )				Student 	Activo 	<input type="checkbox"/>






Figura 3.2: Una de las interfaces para la asignaciones de roles Sakai

**Seleccionar un Rol para los participantes**

Seleccione los Roles de los participantes que está añadiendo. Los campos obligatorios están marcados con \*.

<b>Instructor</b>	Can read, revise, delete and add both content and participants to a site.
<b>Student</b>	Can read content, and add content to a site where appropriate.
<b>Teaching Assistant</b>	Can read, add, and revise most content in their sections.

Usuario	Rol
alejandroperez (Alejandro Pérez)	* Instructor 
danielhernandez (Daniel Hernández)	* Teaching Assistant 




  

Figura 3.3: Una de las interfaces para la asignaciones de roles Sakai

las relaciones entre un identificador de usuario y roles. Los permisos que permiten ejecutar las operaciones: *ver*, *editar*, *borrar* y demás manipulación de datos asociados a las herramientas Sakai, se manejan por medio del uso de *realms* (92). Cada *realm* tiene uno a más roles para asignar a los usuarios. Los permisos son administrados a través de la herramienta *Realms*<sup>c</sup>. Se expone un ejemplo de la estructura de *permisos*, *roles*, *tipo* y *dominios*, que se implementan en una plataforma colaborativa Web. El anexo ?? contiene información ampliada sobre las posibilidades de configuraciones de Sakai. A continuación se muestra un pequeño resumen para ejemplificar lo expuesto. Véase en (91) una descripción completa.

- Un *realm* define los permisos de ciertos objetos de la aplicación
  - Contienen varios roles
  - Cada rol contiene los unos determinados permisos
  - Contiene los usuarios asignados al *realm* y con que rol está asignado

<sup>c</sup>La herramienta Realms de Sakai es utilizada en la administración de permisos y roles para cada uno de los sitios.

- Cada *realm* indica que rol es el de mantenimiento
- Los *realms* se editan con la herramienta de 'Edición de Realms'
- Permite crear nuevos roles
- Permite editar los permisos de los roles
- Permite incluir usuarios
- Los *realms* se editan con la herramienta de 'Edición de Realms'
  - ◇ Permite crear nuevos roles
  - ◇ Permite editar los permisos de los roles
  - ◇ Permite incluir usuarios
- Los roles por defecto de Sakai
  - ◇ !site.template ~> *access, maintain*
  - ◇ !site.template.course ~> *Instructor, Student, Teaching, Assistant*
- Role del creador del Site
  - ◇ Por defecto *maintain, Instructor*
  - ◇ Está especificado por el campo mantenedor del *realm*

#### Modelado del contexto

En esta sección se muestran algunos resultados en base al actual estado del arte que servirán como referentes descriptivos sobre el tipo de modelo de contexto que mejor se adaptaría a las necesidades del DHD. En este sentido, se presenta una referencia a un metamodelo creado para un lenguaje de desarrollo de software guiado por modelos para servicios Web sensibles al contexto basado en UML(93)

#### ContextoDHD: Contexto para los servicios DHD

Un *servicioDHD* es un servicio con sensibilidad al contexto que puede tener cierta flexibilidad para resolver Requerimientos DHD. Esto significa que dichos servicios estarán orientados al tipo de tarea que desempeña un usuario bajo un determinado contexto.

Debido a la heterogeneidad de la información de contexto suministrada, imperfecciones en la transformación de datos, información redundante, malas interpretaciones sobre funcionalidades no computables y la información dinámica del entorno; es imprescindible tener una representación adecuada(94). En particular, varios proveedores de contexto pueden estar reportando las mismas piezas de contexto pero con distintas representaciones. Esto dificulta su especificación en las etapas de diseño.

En los capítulos anteriores se expusieron fundamentos sobre la importancia del contexto en el DHD y consideraciones sobre su visualización funcional a través

### 3.2. El contexto en el DHD

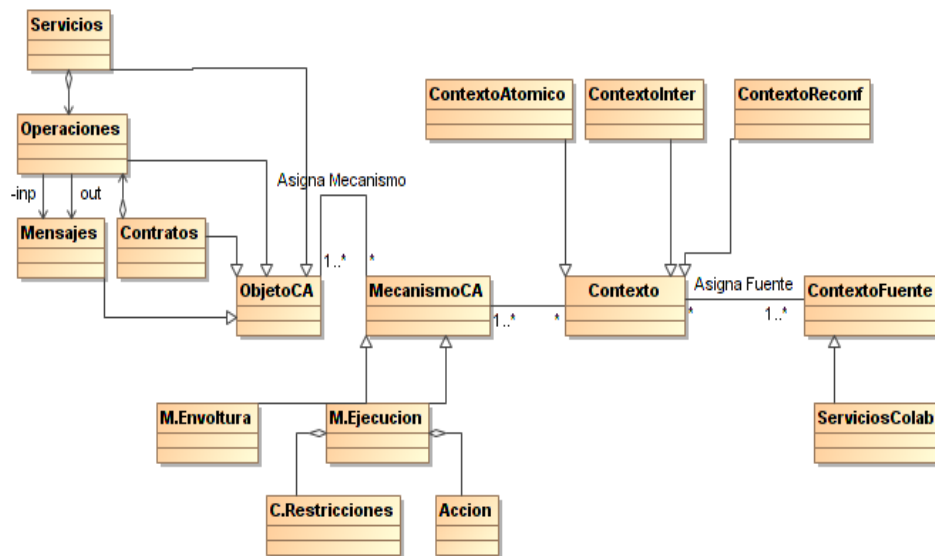


Figura 3.4: Metamodelo del contexto para el DHD

de formas computables de representación. Por este motivo, en esta sección se presentan aspectos de diseño sobre el estado del arte del modelado de contexto que mejor se adapta para las representaciones en el DHD.

En la figura 3.4 se muestra una adaptación del metamodelo creado para ContextUML(?) que fundamenta la creación de una sintaxis abstracta de un lenguaje considerando dos aspectos: *modelado del contexto* y *modelado de los mecanismos de sensibilidad del contexto* (o en inglés *context awareness*).

*Contexto* es la clase que modela la información de contexto. En nuestro diseño adaptado, el tipo de contexto está extendido en dos categorías diferentes como subtipos *CotextoInter*, *ContextoReconf* y *ContextoAtomico*. El contexto atómico es información con el menor nivel de abstracción posible y no comparte relación con otra porción de contexto. Por otra lado, el contexto de las interacciones tiene que ver con información de contexto que colabora en la descripción de las relaciones intersubjetivas. De la misma manera, el contexto para la reconfiguración es una abstracción de los parámetros *context-aware* enunciados en 1.2.1. Estos tres subtipos agrupan a las fuentes del contexto descrito en la figura 3.1, con lo cual ya se cuenta con tres niveles de caracterización del contexto para el DHD. De esta manera, se continúa con similares lineamientos utilizados en el DHD para lograr una representación acorde del contexto hacia su implementación tecnológica.

Es claro observar que desde los servicios se encapsula (oculta) la forma de adquirir el contexto. Así, en las implementaciones donde intervienen servicios de herramientas se usará su misma representación, accediendo a esta información a través de los accesos tecnológicos de la implementación. Por ejemplo, acceso a la bases de datos, acceso a información del servidor Web o del ambiente contenedor

del los lenguajes intervinientes (ej., Apache Tomcat <sup>d</sup>). En otras palabras, se puede decir que el concepto de contexto del servicio oculta la complejidad de su adquisición bajo la perspectiva de los Sistemas Sensibles al Contexto 3.3 (o también mencionados como Sistemas Context-Aware).

**Fuentes de la información de contexto** El tipo de contexto fuente (*ContextoFuente*) caracteriza desde donde parte la construcción del contexto que luego será almacenada para su recolección, por ejemplo, en el registro de actividades. Uno de los subtipos de fuentes pueden ser los servicios colaborativos (representado por *ServiciosColab*) que son utilizados en los servicios de las herramientas colaborativas (ej., Wiki, Foro, etc.). En este caso se tiene una alta dependencia de la implementación de los framework originales (e.i., sin la inyección de contratos sensibles al contexto).

Es observable que desde los servicios se encapsula (oculta) la forma de adquirir el contexto. De esta manera, en las implementaciones donde intervienen servicios de herramientas, se usará su misma representación, accediendo a esta información a través de los accesos tecnológicos de la implementación. Por ejemplo, acceso a la bases de datos, acceso a información del servidor Web o del ambiente contenedor del los lenguajes intervinientes (ej., Apache Tomcat <sup>e</sup>). En otras palabras, se puede decir que el concepto de contexto del servicio oculta la complejidad de la adquisición del contexto, bajo la perspectiva de los Sistemas Sensibles al Contexto 3.3 (o en inglés Context Aware System).

**Contexto para los servicios de reconfiguración** El subtipo representado por la clase *ServicioReconf* interpreta a todos los servicios que devienen de las reglas implementadas por los contratos, mas precisamente, están relacionados con las acciones de estas reglas 5. Por otro lado, los condicionales que forman parte de las mismas reglas se vinculan con la clase *Contexto* haciendo las veces de pre-condición; mientras que los resultados de las acciones establecerán post-condiciones. De esta manera se fundamenta uno de los elementos sobre el por qué de la elección de los contratos como pieza de software base para implementar la reconfiguración y promover la dinámica del DHD.

**Modelado de la sensibilidad del contexto** *MecanismoCA* es la clase que centraliza la formalización de los mecanismos que permiten el modelado para su implementación de las propiedades de sensibilidad al contexto. Para este propósito se han definido dos diferentes categorías por medio de los subtipos *M.Envoltura* y

---

<sup>d</sup>Tomcat es un servidor web con soporte de Servlets y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en Servlets. El motor de Servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

<sup>e</sup>Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

*M.Ejecuciones*. Este mecanismo permitirá determinar la asociación de la información de contexto con objetos concretos que brindan funcionalidades en el sistema. Estos objetos son representados por la clase *ObjetosCA*. Hay 5 subtipos de *ObjetosCA*: Servicios, Operaciones, Mensajes, Partes y Contratos. Cada servicio ofrece varias operaciones y cada operación pertenece a un sólo servicio. Este vínculo se representa por una relación de agregación. Cada operación tendrá un mensaje de entrada y/o uno de salida con la posibilidad de que esos mensajes tengan partes particulares (i.e., mensajes con parámetros). La otra posibilidad es que en vez de mensajes se utilicen, con el mismo propósito, contratos para establecer igual tarea de comunicación con el agregado de mayores propiedades semánticas. Para este caso se establece una nueva relación de agregación entre *Operaciones* y *Contratos*.

**Mecanismos de Envolturas** El subtipo de *M.Envolturas* modela los procesos automáticos de envoltura o vinculación de la información de contexto para que pueda ser procesada por los mecanismos de sensibilidad al contexto. Por ejemplo, en el caso que una operación tome como parámetro un conjunto de caracteres ("string") para identificar un *tema* de un foro, supongamos que un usuario tiene intervenciones en dicho tema del foro con información de contexto referente a su rol (ej., "perteneciente al grupo responsable"). Entonces, el mecanismo envoltura debe vincular el parámetro *tema* con el *rol grupo responsable*.

Una vinculación automática de contexto (e.i., una envoltura de contexto) es una asignación ("mapping") entre un contexto y un objeto sensible al contexto (ej., un parámetro de entrada de una operación de servicio). La semántica se refiere a que un valor del objeto es reemplazado por un valor de contexto. Cabe aclarar que el valor de un objeto sensible al contexto puede derivar varios valores de contexto.

**Mecanismos de Ejecución** El subtipo *M.Ejecución* modela la situación de la adaptación de contexto donde los servicios pueden ser ejecutados o modificados según la información de contexto. Un contexto del mecanismo de ejecución contiene dos partes: un conjunto de restricciones de contexto (*C.Restricciones*) y un conjunto de acciones (*Acciones*), donde la semántica de esas acciones pueden ser ejecutadas si y sólo si se cumplen todas las restricciones impuestas para el contexto.

Las restricciones de contexto especifican que ciertos contextos deben reunir determinadas condiciones con el propósito de cumplir ciertas operaciones. Formalmente, una restricción para el contexto puede estar modelada como un predicado<sup>(93)</sup> (i.e. una expresión booleana) que consiste en un operador y dos o más operandos. En los lugares donde interviene el contrato estas expresiones podrán ser representadas como parte de los condicionales de las reglas. En el caso que esto no ocurra, *M.Ejecución* será el encargado de implementar las restricciones. Un ejemplo de estas expresiones puede ser:

```
tema_foro='\\%DHD\\%' and
```

`id_foro=1233`

para indicar que el operador se ejecutará si se está en un Foro, identificado con el número “1233”, donde el tema tenga el string “DHD”. Esta forma de interpretación posibilita además poner en la misma expresión de acciones para el caso que no se cumpla la condición.

Existe un lenguaje para ContextUML(93) que puede ser usado para este mismo modelo de contexto en la programación de acciones desde el propio mecanismo que implementa la sensibilidad al contexto. En esta tesis, es de interés utilizar solamente el modelo de contexto y todas las acciones de adaptabilidad serán mediadas por la componente contrato.

### 3.3. Sistemas Sensibles al Contexto

En esta sección se toma el antecedente de la representación y formas de utilización del contexto, información de contexto, mecanismos de manipulación y las fuentes de recolección en el DHD. El propósito, es interpretar la posibilidad de agregar a un sistema colaborativo Web (original) las propiedades de sensibilidad del contexto creadas para el DHD (ContextoDHD).

Podemos entender a un sistema colaborativo sensible al contextoDHD como una aplicación provista de mecanismos que permiten una mejor adaptación de los servicios, a partir del contexto de los usuarios y del entorno. En ambientes colaborativos para educación, los servicios forman parte de las funcionalidades observables, desde las perspectivas de los usuarios (ej.:alumno, docente, etc.), que proveen las herramientas del dispositivo (ej., wiki, foros, mensajería, glosario, recursos, etc.). Los elementos que componen la caracterización del contexto deben pertenecer a un dominio bien definido, manteniendo ciertos niveles de concordancia con los mecanismos encargados de manipularlos y la toma de decisiones en base a ellos.

A través de un simple diagrama, es posible observar algunas de las características fundamentales que determinan un sistema colaborativo sensible al contexto, donde se encuentran remarcadas aquellas particularidades (tecnológicas y conceptuales) que se vinculan con la perspectiva del DHD. En la figura 3.5, describimos desde una arquitectura básica y genérica, tres tipos de niveles, cada uno agrega nuevos rasgos y comportamientos que condicionan los servicios del dispositivo hipermedial hacia un nuevo modelo.

La figura está dividida en tres bloques fundamentales, en el primero (Sistema Colaborativos Web Convencional) se describen los principales componentes y relaciones; fundamentalmente un alumno puede interactuar con las herramientas y servicios del sistema y comunicarse con un docente o con sus pares.

### 3.3. Sistemas Sensibles al Contexto

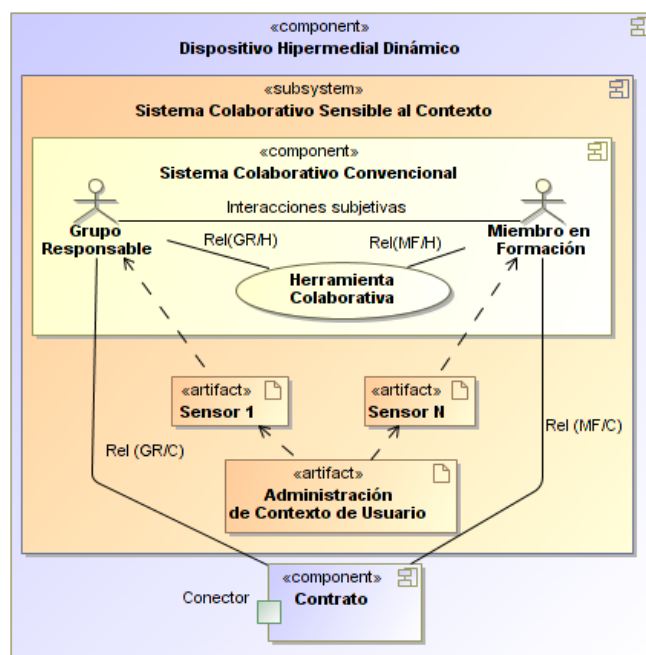


Figura 3.5: Arquitectura propuesta evolutiva.

El segundo bloque de la figura (Sistema Colaborativo Sensible al Contexto) contiene el agregado de dos componentes esenciales, sensores y un administrador de contexto. Los sensores capturan información del usuario y del entorno, son partes fundamentales para la recolección de información; si los sensores juegan un papel secundario, los podemos enmarcar bajo el concepto de artefactos virtuales. El segundo componente hace referencia a la administración del contexto, donde se destacan las siguientes funcionalidades: recolección, abstracción, interpretación, comunicación y almacenamiento. Para la implementación de comportamientos context-aware los diseños de software deben completarse con diferentes modelos, para el caso de los denominados sistemas e-learning, las soluciones se focalizan en la tecnología de comunicación e información que utilizan los usuarios. MatchBase (Gross et al., 2006) es un proyecto de referencia en cuanto al uso de estos tipos de diseño y tecnología, conformando una herramienta para el desarrollo de comunicaciones contex-aware. Fue trasladado como experiencia a proyectos de investigación de sistemas e-learning context-aware (Schmidt, 2005). Pensar en aplicaciones para educación e investigación sensibles al contexto dentro del marco teórico context-awareness conforma un nuevo paradigma, donde las relaciones (rotuladas con "Rel (fuente/destino)" en la figura) cobran un mayor protagonismo en la concepción de los marcos conceptuales propuestos como soluciones a nuevos requerimientos académicos para el contexto físico-virtual.

Perspectivas como la del DHD, también centran los requerimientos sobre las relaciones y comunicaciones entre los principales componentes del dispositivo, focalizados en las relaciones entre los actores (Recursos Humanos en formación, grupos responsables) y los servicios brindados por las herramientas anteriormente mencionadas, a las que se les integra el potencial de comunicación hipermedial como por ejemplo, video conferencias, edición en variados formatos, etc.

Retomando la evolución de los sistemas colaborativos referida a la adaptación de modelos para requerimientos educativos complejos, el último bloque de la figura (Dispositivos hipermediales sensibles al ContextoDHD Dinámico) representa un modo de interpretación de una nueva configuración de la arquitectura de un sistema, debido a la interposición de la componente contratos (referenciada con el dibujo característico para componentes de software en UML). Al agregar este nuevo elemento que permite una mejor articulación de las relaciones, resulta una nueva teoría que proporciona conceptualmente un marco innovador para las prácticas de diseño, desarrollo, uso de dispositivos y aplicaciones en dicho campo.

El contrato debe ser visto como una alternativa de abstracción de las relaciones mencionadas anteriormente, determina un nuevo tipo de relación que mantiene las características de los anteriores bloques de la figura y además, reduce la complejidad de los modelos de los SSC en la adaptación de los servicios que interfieren en relacionamientos entre usuarios y comunicacionales. Los contratos se nutren con información del contexto por medio de sus parámetros; no intervienen en la recolección, ni en la abstracción, ni en la distribución del contexto y en principio pueden ser adaptados a cualquier modelo de sensibilidad al contexto similar, bajo la perspectiva propuesta por Dey. et. al (2001).

A continuación, se analizan las principales características de los SSC. Como observamos en la figura anterior, el bloque intermedio se configura como articulador entre los sistemas colaborativo tradicionales y las nuevas posibilidades tecnológicas que permiten la concreción más efectiva del Dispositivo Hipermedial Dinámico agregando la componente Contrato.

Esta propuesta evolutiva y la forma de representación del contexto mantiene los principios que definen la teoría de los SCC.

El contexto puede definirse de manera general como:

Cualquier información que puede usarse para caracterizar la situación de una entidad. Donde una entidad es una persona, lugar u objeto que es considerado relevante para la interacción entre un usuario y una aplicación, incluyendo al usuario mismo y la aplicación

(Dey y Abowd, 2000)

Las aplicaciones sensibles al contexto se definen como:

Aquellas que usan el contexto para proveer información y/o servicios relevantes al usuario, donde la relevancia depende de la tarea del usuario.

(Dey y Abowd, 2000)



### 3.3. Sistemas Sensibles al Contexto

---

Según un análisis realizado por Brown et al., (2000), entre las aplicaciones sensibles al contexto, la recuperación de información juega un papel central, y tales aplicaciones parecen confirmarse como la prospectiva que requiere el computo consiente de contexto.

Por lo tanto, en las aportaciones de los autores de los trabajos que presentaremos a continuación, tomaremos el problema planteado en el marco de la recuperación de información consiente de contexto.

#### 3.3.1. Propuestas para la Identificación del Contexto

Algunos trabajos se han enfocado sobre cómo identificar el contexto efectivamente y cómo obtener una representación del mismo que pueda ser utilizada para incorporarlo a la búsqueda de información relevante para el usuario.

Budzik y Hammond, (2000) clasifican los esfuerzos llevados a cabo para esta tarea en cuatro categorías:

- **Retroalimentación sobre la relevancia de los resultados:** Trata de determinar el contexto obteniendo cierta retroalimentación por parte del usuario respecto al nivel de relevancia de los resultados que pueda tener una consulta (Salton y Buckley, 1990).
- **Perfiles de usuario:** en este caso los intereses del usuario se integran para formar un perfil que es efectivo a lo largo de varias consultas, un ejemplo de este enfoque es Letizia (Lieberman, 1995).
- **Eliminación de ambigüedad en el sentido de las palabras:** Algunos sistemas tratan de eliminar posibles ambigüedades pidiendo al usuario que lo haga explícitamente (Cheng y Wilensky, 1997), mientras que otros, lo hacen implícitamente usando la popularidad de los términos según la estructura interna de los documentos de hipertexto (Bradshaw y Hammond, 1999).
- **Enfoques de ingeniería del conocimiento:** intentan crear un modelo de comportamiento del usuario mientras estos se encuentran interactuando con una aplicación.

A su vez, Lawrence (2000) propone cinco tipos de estrategias para incluir el contexto en las búsquedas:

- **Agregar información de contexto en forma explícita:** se presentan al usuario mecanismos para que éste especifique el contexto en el cual desea realizar su búsqueda. Tal es el caso del proyecto Inquirus 2 (Glover et al., 2000).

- **Inferir la información de contexto automáticamente:** los sistemas tratan de obtener dicha información de contexto sin la intervención conciente del usuario, usualmente monitoreando las actividades y aplicaciones de éste. El proyecto Watson (Budzik y Hammond, 2000) se inscribe en esta categoría. Siguiendo este mismo enfoque hay sistemas que recomiendan enlaces a sitios Web dinámicamente mientras el usuario realiza sus tareas, algunos ejemplos de este tipo de sistemas son: The Remembrance Agent (Rhodes, 2000), SurfLen (Fu et al., 2000), Margin Notes (Rhodes, 2000), y Fab (Balabanovic, 1997).
- **Búsquedas personalizadas:** este concepto implica que una máquina de búsqueda conozca todas las búsquedas anteriores del usuario y que use esa información para personalizar los resultados de búsquedas futuras. Con el buscador Northern Light el usuario recibe notificaciones sobre nuevas páginas que concuerden con ciertas expresiones de búsqueda.
- **“Adivinar” lo que el usuario quiere:** este enfoque consiste en examinar la expresión de búsqueda del usuario y cuando se encuentran ciertos términos predefinidos el buscador arroja en los resultados ciertas páginas asociadas a tales términos. Google ([www.google.com](http://www.google.com)) percibe una cadena de caracteres que tiene el formato de una dirección, arroja resultados con enlaces a mapas de esa zona. Otros ejemplos son ([www.excite.com](http://www.excite.com)) y ([www.lycos.com](http://www.lycos.com)).
- **Restringir el contexto de las máquinas de búsqueda:** otra manera de delimitar el contexto es restringir el dominio de la misma máquina de búsqueda.

Actualmente disponemos de gran diversidad de buscadores especializados, algunos ejemplos son: CiteSeer (Lawrence et al., 1999) y Deadliner (Kruger et al., 2000). También existen meta buscadores que implementan algún mecanismo para derivar el contexto del usuario y luego usan esa información para realizar consultas sobre alguno de estos buscadores especializados. Tal es el caso de Inquirus 2 (Glover et al., 2000). Sobre los enfoques para “Derivar el contexto pasado y futuro”, Brown y Jones (2002), proponen explotar la característica del contexto que lo hace cambiar constantemente pero en cierta medida de manera predecible.

Al analizar cómo es el proceso de cambio y llevar un registro del mismo, podemos crear lo que dichos autores llaman: “Diario de Contexto”, guardando los estados de las variables de contexto medidas por sensores ya que a través de esta información se puede predecir un contexto de interés a futuro. También proponen un “Context-aware Caching” (depósito de documentos) en el cual, se recupera la información, haciéndolo con un contexto en donde los campos (o variables) son rangos estimados de valores que puede tomar cada campo en un futuro cercano, utilizando los documentos resultantes para formar un depósito temporal desde el cual se pueden realizar mas rápidamente recuperaciones posteriores mientras el contexto del usuario no cambie considerablemente.

#### 3.3.2. Visiones en la perspectiva histórica

Es interesante considerar en el marco de esta tesis, algunos de los antecedentes más relevantes a nivel histórico del paradigma que subyace en los sistemas sensibles al contexto, citaremos a continuación destacados referentes:

**Vannevar Bush's Memex (1945)** Propone, antes del desarrollo del computador, un aparato denominado Memex que como suplemento de nuestra memoria, facilitaría el acceso y la relación de la información acumulada. La clave de este dispositivo, es que funcionaría imitando los procesos de asociación de la mente humana. Reconocido el límite que impone el artificio, se plantea igualmente un cambio fundamental: la selección debe ser por asociación y no por la ubicación mecánica de temas en un índice alfabético.

En el Memex se podrían guardar archivos, libros y textos para ser consultados con rapidez y flexibilidad, sería posible agregar comentarios y notas marginales. Quien consultara, construiría senderos de lectura de acuerdo a su interés, seleccionando y enlazando los artículos a través del laberinto de materiales disponibles, y podría modificar esta configuración cuando lo deseara. Bush concretiza a nivel de objeto tecnológico la recuperación y escritura de la información en formato hipertextual. Edward Thorp: "The invention of the first wearable computer" (1960) La primera computadora "wearable" (que se puede llevar puesta en el cuerpo) fue concebida en 1955 por Edgard Thorop para predecir el comportamiento de la ruleta, culminando en un trabajo conjunto en el M.I.T. con Claude Shannon en 1960-61. El dispositivo era similar al tamaño de un paquete de cigarrillos y permitía aumentar hasta un 44 % las chance de ganar, si bien se probó el objetivo planteado, luego ocurrieron pequeñas fallas en el hardware que impidieron continuar con el uso de este dispositivo. Este método y la existencia de la computadora no fueron publicados hasta el año 1966.

**Alan Kay's Dynabook (circa 1977)** La Dynabook es un dispositivo portátil, con red inalámbrica y pantalla plana, entre otras cosas. Este dispositivo se concibe como extensión de las posibilidades de la mente-cuerpo y lugar donde un usuario concentra toda la información que consume y que genera. Guiados por esta visión, un grupo de referentes de la informática (Alan Kay, Dan Ingalls, Adele Goldberg, etc.) fueron responsables de los desarrollos de mayor impacto relacionados con las computadoras personales. Una lista, no completa, de los aportes de este proyecto son: "El concepto de la Computadora Personal", "El paradigma de objetos", "Smalltalk", "Interfaces de usuario gráficas", "Uso del Mouse", "Drag drop", "Menús despleables".

Las ideas de la Dynabook siguen vigentes en el Squeak y su filosofía en grupos como SqueakLand. Los eToys y los Ensayos Activos son subyacentes a las ideas del meta-medio.

**Mark Weiser; “The Computer for the 21st Century (1991)** La computación ubicua como paradigma de interacción fue introducida por Mark Weiser en 1991 después de sus trabajos en los laboratorios de Xerox PARC en Palo Alto (Weiser, 1991). La computación ubicua pretende ampliar la capacidad computacional a todo el entorno mediante la distribución de pequeños y muy diversos dispositivos que presentan ciertas características interactivas, todos ellos conectados a servidores de mayor potencia.

El diseño y situación de estos dispositivos debe estudiarse rigurosamente, según la tarea que realizan. De este modo, la responsabilidad computacional se desplaza diluyéndose en el entorno (transparencia del objeto computacional), intentando suscitar la idea de omnipresencia (Norman, 1998). La solución propuesta por Weiser (1991-1998) fue disponer de redes inalámbricas para computadoras con la finalidad de intercambiar información entre ellas y configurar un dispositivo de interacción. En los ambientes ubicuos hay tres tipos de computadoras: “marcas”, “tabletas” y “pizarras”.

En el Centro de Investigación de Xerox de Palo Alto, diseñaron estas computadoras, (UbiCom) y Weiser en 1998, sostuvo que su propuesta podría tener mayor aceptación en los campus universitarios. En 1999, distintos equipos de investigación adoptan internacionalmente con fines educativos este nuevo paradigma para el aula. Soloway et. al (1999) expuso el beneficio que esto significaría para el aprendizaje por descubrimiento, mediante la realización de experiencias que surgen a partir de los principios del paradigma de computación ubicua, al considerar estos dispositivos como controles remotos de una pizarra de grupo. El mencionado investigador, propone diseñar dispositivos y periféricos que sirvan para captura de datos en el entorno real mediante la utilización de dispositivos handheld. Posteriormente estos datos serían enviados a un servidor que los presentará para su discusión grupal.

**Bill Schilit: “Context-Aware Computing Applications” (1994)** Schilit define computación context-aware por medio de la caracterización de aplicaciones context-aware de la siguiente manera:

Selección próxima: una técnica de interface de usuario donde el objeto más próximo es señalado o facilitado para su mejor selección.

- Reconfiguración automática de contexto: es el proceso por el cual se agrega un nuevo componente, se quita un componente existente, o se altera la conectividad entre dos componentes ante un eventual cambio en los elementos que componen el contexto.
- Información de Contexto y comandos: los cuales pueden producir diferentes resultados de acuerdo al contexto que donde se utilizan.
- Lanzamiento de Acciones ante cambios de Contexto: son simples reglas tipo IFTHEN utilizadas para especificar cómo se debe comportar el sistema context-aware.

### 3.4. Componentes bases para los Sistemas Sensibles al Contexto

---

**Don Norman: “The invisible Computer” (1998)** A partir de los años '80, con la difusión masiva de las interfaces user-friendly, se consolida entre muchos proyectistas e investigadores una concepción que privilegia una lectura sobre la interacción hombre-computadora en términos instrumentales y que tiende a hacer “desaparecer” la interfaz. Don Norman, propone que los mejores programas informáticos “son aquellos donde la computadora ‘desaparece’ y se puede trabajar sin tener en cuenta a la máquina.” (1989:231). Esta aparente invisibilidad de los procesos de interacción.<sup>es</sup> una consecuencia directa de la aplicación de la metáfora mcluhaniana sobre la extensión de las interfaces digitales como prótesis de nuestro cuerpo. Esta perspectiva, propone que la mejor interfaz es aquella que desaparece durante el uso. Según Anceschi “las interfaces deberán ser lo más transparente posible, o sea, deberían tener la menor consistencia (perceptiva) posible” (1993:19). Gui Bonsiepe, sostiene que “el usuario ha aprendido el uso de un programa cuando éste se vuelve tan transparente que ya no tiene necesidad de ‘pensar’, o sea cuando el programa desaparece y el usuario puede ocuparse de la ejecución de la tarea que se propone realizar...” (1993b:52). De Kerckhove, afirma que antes de alcanzar el nivel de saturación una tecnología debe atravesar dos fases: en la primera el dispositivo resulta muy evidente, en la segunda se interioriza “hasta volverse invisible” (De Kerckhove, D.:1999, 123).

### 3.4. Componentes bases para los Sistemas Sensibles al Contexto

En esta sección, expondremos los diferentes tipos de componentes para capturar contexto que integran un Modelo para la Sensibilizarlo al Contexto (MSC). Particularmente, mencionaremos dos tipos de moles emblemático con diferentes enfoques.

#### 3.4.1. Context Toolkit

En primer lugar, se analiza un modelo tradición fundador expuesto en la tesis doctoral de Dey (2003)(? ). La siguiente tabla muestra una clasificación de arquitecturas en las que se resaltan las principales características:

- 
- Acceso directo a Sensores:
    - No extensible
    - Componentes altamente acopladas
- 
- Basado en Middleware:
    - Permite la ocultación de detalles de censado de bajo nivel.
    - Extensible.
-

### 3.4. Componentes bases para los Sistemas Sensibles al Contexto

---

- Permite el acceso de múltiples clientes a los datos de forma remota.
  - Libera a los clientes de recursos que demandan
- 

- Servidor de Contexto: operaciones intensivas.

- Debe considerar el uso de protocolos apropiados, rendimiento de la red, calidad de los parámetros de los servicios.
- 

- Widgets:

- Encapsulamiento.
  - Intercambiable.
  - Controlado a través de un manejador de widget.
  - En el caso que las componentes estén acopladas incrementan la eficiencia, al costo de perder robustez ante fallas de componentes.
- 

- Networked services: (modelo orientado a objeto)

- Similar a la arquitectura Servidor de Contexto.
  - No tiene la eficiencia de la arquitectura widget debido a la complejidad de las componentes bases de red, pero provee robustez.
- 

- Blackboard model: (modelo basado en datos)

- Procesos pos-mensajes para compartir medios, pizarrón.
  - Simplicidad en el agregado de nuevos recursos de contexto.
  - Fácil configuración.
  - Un servicio centralizado.
  - Carece de eficiencia en la comunicación (son necesarios dos puntos de conexión por comunicación).
- 

Llegado a este punto, introduciremos una definición de la componente principal del modelo: el Widget.

Este componente, deriva del homónimo elementos GUI, un Widgets es un componente de software que brinda una interface pública para algún tipo determinado de sensores (hardware) (Dey and Abowd, 2000, 2001). Los Widgets ocultan detalles de censado de bajo nivel y al mismo tiempo son componentes con alto grado de reusabilidad, facilitando el desarrollo de aplicaciones. El encapsulamiento en los widgets permite intercambiarlos entre aquellos que proveen el mismo tipo de datos (ejemplo: intercambiar un widget de radiofrecuencia por un widget de una cámara filmadora donde ambos recolectan datos de locación de individuos).

Usualmente los widgets son controlados por alguna clase de administrador de widget. En el caso que las componentes sean acopladas incrementan la eficiencia, al costo de perder robustez ante fallas de componentes. Se identifican cinco categorías de componentes que implementan distintas funciones:

### 3.4. Componentes bases para los Sistemas Sensibles al Contexto

---

- **Context Widgets:** se encarga de la adquisición de información de contexto.
- **Intérpretes:** cumplen la función de transformar y aumentar el grado de abstracción de la información de contexto; deben combinar varias piezas de contexto para producir información procesada de alto nivel.
- **Aggregator:** es un componente que junta información de contexto de una entidad para facilitar su acceso a las aplicaciones.
- **Services:** brindan servicios en un determinado ambiente adaptando su funcionalidad al tipo de información de contexto adquirida.
- **Discoveres:** permite que las aplicaciones (y otras entidades) optimicen su desempeño pudiendo determinar la característica del entorno (tipo de restricciones y dominio de aplicación). Entre estos componentes se pueden establecer un número limitado de relaciones.

Widgets es consultada, o bien, notificada ante eventuales cambios en los clientes. Los clientes pueden ser aplicaciones, aggregators u otros widgets. A su vez aggregators actúa como un puente entre widgets y aplicaciones. Un interpreters puede ser solicitado en un determinado estado por un widget, aggregator o aplicaciones. Los services son lanzados por las aplicaciones (también otros componentes pueden hacer uso de los servicios). Discoveres se comunica con todos los componentes, adquiere desde los widget, interpreters y aggregators, y provee información a las aplicaciones por medio de notificaciones y consultas.

La figura 3.6 expone una posible configuración con dos dispositivos de censo, dos widgets, un aggregator, dos interpreters, un servicio, un discoverer y dos aplicaciones.

En el instante en que algún componente contexto se encuentre disponible, registra sus capacidades en un discoverer. Esto permite que los *aggregators* encuentren *widgets* e *interpreters* y a las aplicaciones encontrar *aggregators*, *widget* e *interpreters*. Un sensor provee datos para un *context widget*, en cual se encarga de almacenar el contexto, puede llamar a un *interpreter* para obtener un mayor nivel de abstracción de los datos y luego pone el contexto a disposición (para que pueda ser accedido) por otros componentes y aplicaciones. Un *aggregator* recolecta información de contexto de las entidades, representadas por los *widgets*. Finalmente, las aplicaciones pueden consultar o suscribirse a los *agregators* (o directamente con los *widgets*, si se quiere) y llamar a *interpreters* (si el deseado nivel de abstracción no se encuentra disponible desde los *widgets* y *aggregators*).

Estos componentes se ejecutan independientemente de las aplicaciones, asegurando una continua adquisición de información de contexto y el uso de múltiples aplicaciones. Además, todos los componentes y aplicaciones se comunican entre ellos automáticamente usando protocolos y lenguajes conocidos. Esto da lugar a que los programadores que implementaron un particular componente o aplicación puedan comunicarse con otro componente sin tener conocimiento de los mecanismos usados para lograr la interacción.

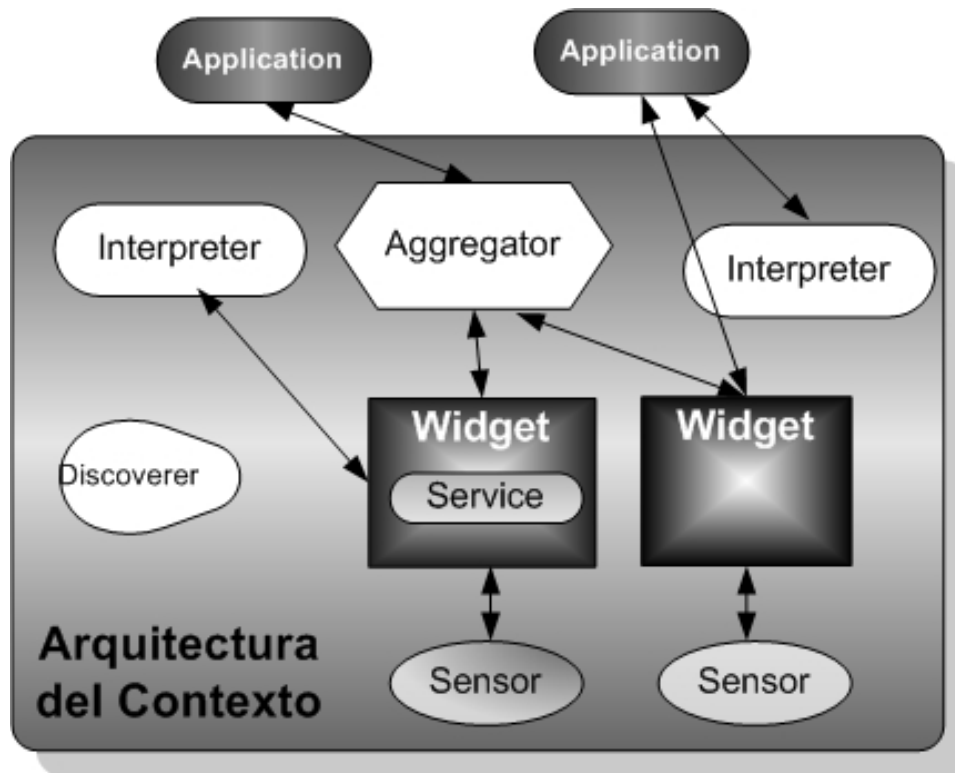


Figura 3.6: Una configuración de las componentes del context Toolkit.

#### 3.4.2. Proyecto UWA

UWA ha nacido de la colaboración entre diferentes grupos de trabajo, por lo que resulta realmente una agrupación de propuestas y técnicas. UWA incluye una original propuesta de aplicar la notación UML para modelar elementos de multimedia que a su vez es heredado de otra propuesta llamada HDM (Hypermedia Design Model)(27). Sin embargo, W2000 ha sido incluida en UWA sólo en la fase de diseño hipermedia, siendo ambas propuestas diferentes en la fase de definición de requisitos. Por esta razón han sido incluidos en este proyecto en forma separada. El proceso de captura de requisitos en UWA(?) comienza definiendo los diferentes roles de usuario que pueden interactuar con el sistema, los objetivos globales del sistema y la relación entre éstos. El proceso continúa haciendo un refinamiento de esos objetivos globales, concretándolos en subobjetivos. Estos subobjetivos son estudiados y refinados para detectar conflictos entre ellos. De esta forma, se concretizan aún más dividiéndolos en requisitos. Los requisitos son clasificados en varios tipos: de *contenido*, de *estructura de contenido*, de *acceso*, de *navegación*, de *presentación*, de *operaciones de usuario* y de *operaciones del sistema*. De esta forma, los requisitos se van refinando hasta que solo pertenezcan a uno de estos grupos. Y finalmente los requerimientos son asignados a artefactos de diseño o a reglas de customización. Para definir los objetivos, UWA propone una notación propia, basada en una plantilla. La definición de los actores y la relación con los objetivos se hace usando un diagrama basado en casos de uso. Por último, para definir y refinar los subobjetivos y los requisitos, utilizan una notación



### 3.4. Componentes bases para los Sistemas Sensibles al Contexto

---

gráfica propia que denominan grafo de refinamiento de objetivos, el refinamiento de este grafo permite ir representando la relación entre los requisitos y hacer un seguimiento para validar la consecución de los objetivos del sistema. Una vez que los requisitos son detectados, hacen uso de XML para definirlos de una manera formal.

"An ubiquitous web application is a Web application that suffers from the anytime/anywhere/anymedia syndrome. This means that an ubiquitous web application should be designed from the start taking into account not only its hypermedia nature, but also the fact that it must run "as is" on a variety of platforms, including mobile phones, Personal Digital Assistants (PDAs), full-fledged desktop computers, and so on. This implies that an ubiquitous web application must take into account the different capabilities of devices comprising display size, local storage size, method of input, network capacity, etc. New opportunities are offered in terms of location-based, time-based, and personalised services taking into account the needs and preferences of particular users. Consequently, an ubiquitous web application must be, on the one hand, context-aware i.e., aware of the environment it is running in, and on the other hand it must support personalisation. From: Ubiquitous Web Application Development- A Framework for understanding"

*A. Finkelstein et al 2002*

Se comienza con la descripción conceptual del framework propuesto en el proyecto UWA para el desarrollo de aplicaciones web ubicuas basada en el enfoque de la teoría de la reflexión (*reflection*)

**Goal** Goal es un objetivo que el sistema debe alcanzar por medio de la cooperación entre agentes (usuarios y componentes software) propiamente dicho y dentro de un entorno. Desde la perspectiva UWA los objetivos son *inmutables* i.e., no cambia ante cuando se modifica el entorno. Representa el último objetivo que el servicio está destinado a lograr. Un cambio en los objetivos significaría cambiar el servicio mismo. A lo largo de las líneas de (?), donde un goal no es alcanzado por una acción directa de uno a más agentes; de otro modo, un goal es un objetivo un tanto abstracto de largo plazo.

En la Figura 3.7, los objetivos son de color gris, lo que significa que ellos son la única parte del marco que no tiene la intención de cambiar en tiempo de ejecución. Más exactamente, las metas pueden (y deben) tener una imagen en tiempo de ejecución (ya que su puesta en marcha en los requisitos de forma dinámica puede cambiar), pero no se pueden modificar directamente.

**Service** El servicio se puede definir como algo que proporciona valor añadido a uno o más actores. Un servicio es distinto de una aplicación, ya que es un con-

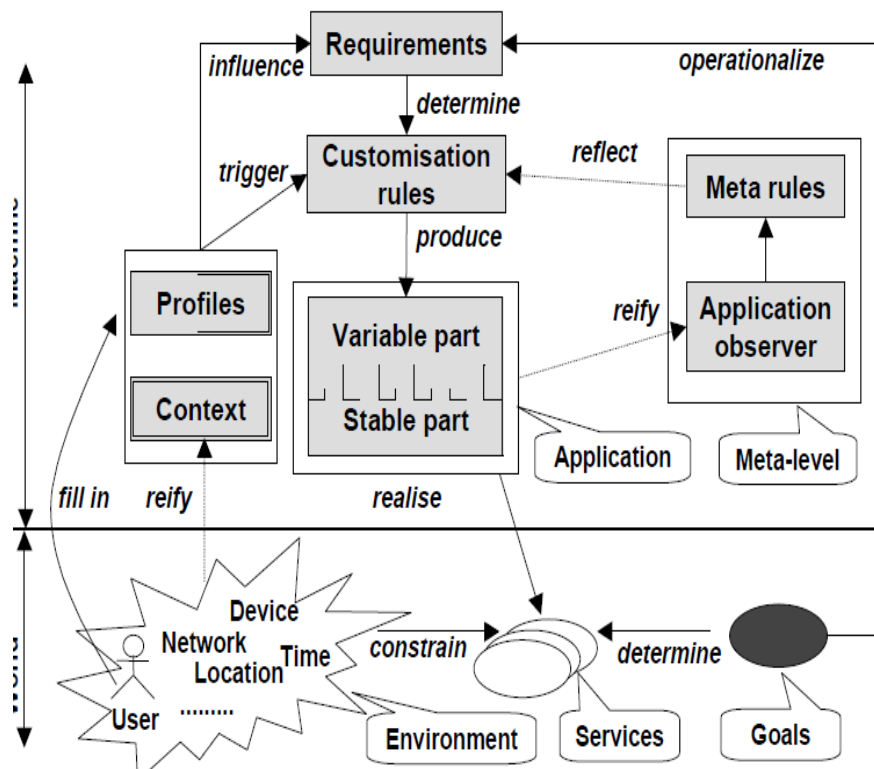


Figura 3.7: Diseño conceptual del Framework UWA

cepto mucho más general, que podría ser proporcionada por ejemplo, por medios distintos de computadoras. Por ejemplo, si uno de los objetivos es maximizar la usabilidad del sistema, un servicio puede ser el de proporcionar a los usuario con los dispositivos de entrada-salida. Este es un ejemplo de diseño centrado en el usuario, mediante el cual el sistema es sólo un medio para cumplimiento de las metas de usuario, y no un objetivo en sí mismo. Por lo tanto, los servicios de pertenecer en el mundo, no en la máquina.

**Environment** Por "Environment" se refiere al entorno en que la máquina operar. Teniendo en cuenta el medio ambiente es crucial porque influye mucho en el comportamiento de la máquina.

El entorno cuenta con propiedades para describir una faceta diferente del medio ambiente. Consideremos el ejemplo de un servicio de m-commerce. En este caso el medio ambiente abarca cosas tales como ancho de banda, la ubicación (absoluto y relativo), la disponibilidad del servicio, las características del dispositivo, y muchos temas más. Tenga en cuenta que el sistema no tiene control alguno sobre el medio ambiente. En otras palabras, la máquina debe adaptarse a las medio ambiente, no al revés. Como cuestión de hecho, si el ancho de banda es bajo, la conexión es irregular, el PDA pantalla es pequeña, esto es algo que no puede ser modificado directamente por el software. El trabajo de un ingeniero de software por lo tanto puede resumirse como una lucha hacia la meta a pesar del entorno, todo lo que puede hacer con el medio ambiente es sentido, y lo describen de la mejor manera

### 3.4. Componentes bases para los Sistemas Sensibles al Contexto

---

posible, pero no podemos cambiar i

**Context** Contexto se define como la resignificación del medio ambiente en términos de sus propiedades. En la figura, desde el contexto no salen flechas hacia el entorno medio ambiente ya que UWA lo interpreta como no modificable. A su vez, el contexto permite efectuar una descripción del medio ambiente. La mayoría de las descripciones importantes deben ser dinámicamente actualizables en función del cambio posible del medio ambiente.

El contexto es parte de la máquina, ya que es una representación del medio ambiente (que a su vez pertenece al mundo real) dentro del sistema. En otras palabras, el medio ambiente se representa en el contexto, ya que existe una máquina; sin esto, el contexto no tendría sentido. Tenga en cuenta que el contexto es la única parte del sistema que es la aplicación independiente, en el que se describen las propiedades del medio ambiente adecuado, independientemente de la aplicación que se va a construir. Esto está marcado en la Figura 3.7 por un doble contorno alrededor de la caja de contexto.

**Profiles** A diferencia del contexto, el perfil representa información explícita. Un ejemplo es el uso de perfil para la personalización de usuarios. El perfil puede ser dependiente o independiente de la aplicación.

**Requirements** Los Requerimientos representan una de las posibles formas de lograr una meta. Un requerimiento personaliza un objetivo, ya que representa en una forma más concreta los objetivos a corto plazo que son directamente alcanzables a través de las acciones realizadas por uno o más agentes. Los requisitos pueden cambiar durante la ejecución del sistema. El entorno puede sufrir un alto grado de cambio invalidando los objetivos iniciales. Entonces, cambios en el contexto puede generar la necesidad de cambios en los requerimientos.

En términos informal, se puede decir que los requisitos son un "trade-off" entre los objetivos y la realidad actual. Por ejemplo, el objetivo de una aplicación m-commerce ubicua podría ser establecida por la alta interactividad de un usuario. Teniendo en cuenta este objetivo, si el contexto es favorable (por ejemplo, el ancho de banda, alta, color de gran tamaño pantalla, máquina virtual Java disponible) un requisito puede ser utilizar un colorido applet de Java para representar el estado de la cesta de la compra", mientras que si la conexión es lenta o no hay JVM disponible, el requisito puede ser mitigado en utilizar un gif animado de 16 colores."

**Application** La aplicación es la encargada de la implementación de uno o más servicios. La relación entre aplicación y servicios es de mucho a muchos. Un servicio puede estar relacionado con varias aplicaciones y una aplicación puede disparar varios servicios. Un servicio complejo puede estar interactuando con varias aplicaciones distribuidas.

Una aplicación está conformada por una *parte estable*, que provee funcionalidades básicas y una *parte variable*, que se encarga de efectuar las adaptaciones. Ambas partes están vinculadas por medios de "ganchos" que implementan la comunicación.

La solicitud se hace por una parte estable, que proporciona la funcionalidad básica, y una parte variable, que se da cuenta de la personalización. La parte estable de la aplicación proporciona los ganchos, los llamados puntos calientes de personalización, que permite la parte variable para adaptar su comportamiento a las necesidades particulares. Bajo este aspecto, se asemeja a una orientada a objetos marco.

**Customisation Rules** Las reglas de personalización (Customisation Rules) son el medio por el cual los requisitos se traducen en la parte variable de la aplicación y se puede conformar como reglas de producción.

**The Meta Level** El Meta Nivel (Meta level) es una representación de la aplicación en términos reflectivos. Implementa las eventuales conexiones de la aplicación, esto se refiere al estado de la aplicación, visto de otra perspectiva, funcionan como un continuo monitoreo y mantiene una descripción actualizada de su estado. Esto es necesario ya que el funcionamiento de la aplicación puede ser influenciada por el medio ambiente, y por lo tanto puede cambiar de maneras impredecibles. Supongamos, por ejemplo, que un teléfono móvil está funcionando en un área con fuertes campos magnéticos. En esta situación, el teléfono puede reportar un buen conexión, pero la calidad real siempre puede ser baja debido a la interferencia. En tal caso, el control de la medio ambiente no es suficiente, y no hay manera de darse cuenta de esta interferencia que no sea el control de la propia aplicación. La relación con la aplicación se produce mediante dos vías, donde el *meta nivel* puede afectar el nivel de base funcionalidad por medio de la *meta reglas* que permitan implementar la personalización correspondiente. Entonces, ante situaciones particulares las metas reglas son disparadas y pueden cambiar, añadir o eliminar las reglas actuales

Un meta nivel de reflexión también se puede emplear para los requerimientos de control, en el sentido utilizado en (?).

en este nivel se implementa una de las principales características que se pretende poner en obra en esta tesis debido a su relación con nuestra visión de reconfiguración dinámica.

Se argumenta que la "manera reflexiva" es una manera limpia y consistente de los cambios de rendimiento en tiempo de ejecución a la nivel subyacente (que es finalmente el sistema real como percibida por el usuario). Este enfoque consiste en la manipulación las normas de personalización con el fin de conciliar el servicio con los requisitos. La relación de causalidad, en particular, el enlace descendente (reflexión), prevé la coherencia entre la descripción del servicio y el propio servicio.

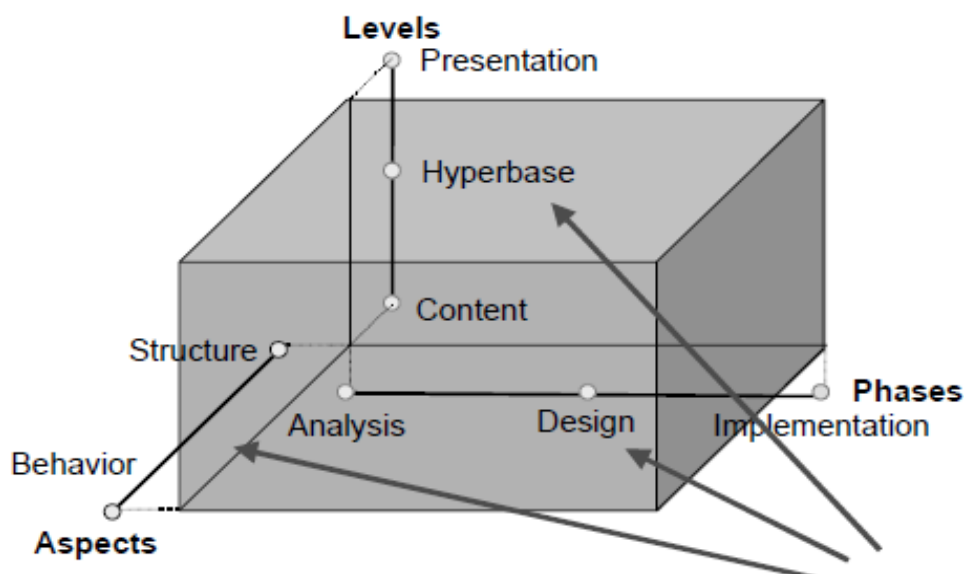


Figura 3.8: Diseño conceptual del Framework UWA

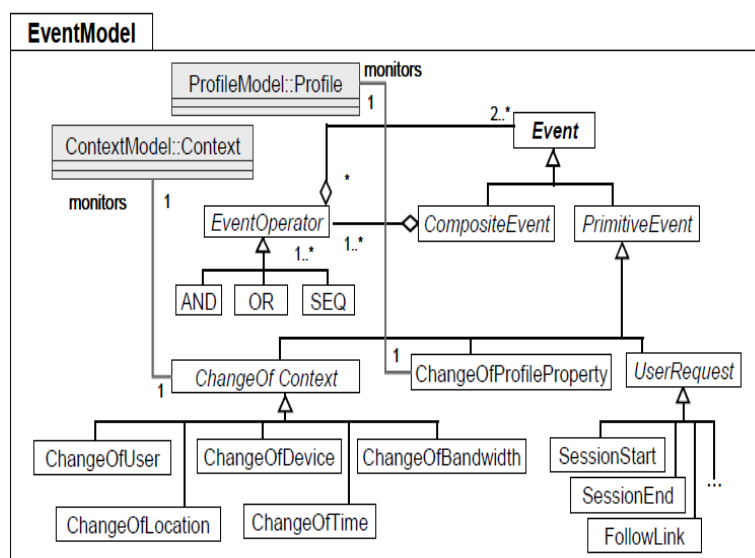


Figura 3.9: Diseño conceptual del Fram

Técnicas de arquitectura reflexiva puede ser empleado a tal fin [23].

### 3.5. Hacia la Ingeniería de Software

En este capítulo se han explicitado los fundamentos y desarrollos básicos de la evolución de los SCC. Introducimos la necesidad de que los mismos tuvieran un nivel dinámico sustentado en la teoría de coordinación de contratos, ya que las

definiciones más tradicionales no son suficientes para satisfacer los requerimientos del DHD.

En los capítulos siguientes, se abordará lo inherente a los contratos como pieza de software en base a la arquitectura, el diseño y la implementación en el campo de la Ingeniería de Software para describir las problemáticas, motivaciones, el tipo de solución propuesta y un marco de conceptualización-definición (capítulos: 1 y 2 ) en un contexto de alta complejidad como lo ya expuesto, para tratar requerimientos funcionales derivados del mismo.

Se tomarán en cuenta todas las convenciones realizadas sobre el DHD, haciendo foco en adelante sobre lo específicamente tecnológico estableciéndose la intersección interdisciplinar entre las competencias de los usuarios expertos y los ingenieros de software como destinatarios principales de este trabajo de tesis doctoral.

Es importante para el desarrollo de los próximos capítulos tener presente los siguientes interrogantes que se constituyen en el sentido sustentador para el uso de la teoría de Sistemas Colaborativos Sensibles al Contexto en las aplicaciones que integran al DHD.

Preguntas
¿Cuáles son las variables que definen el contexto del usuario que pueden ser usadas para recuperar información relevante?
¿De qué forma puede modelarse el contexto del usuario?
¿Cómo debe incorporarse el contexto a la búsqueda del usuario?
¿Cómo deben presentarse los resultados derivados del contexto del usuario?
¿Cuál es la relación costo/beneficio de utilizar el contexto en la recuperación de información?
¿De qué forma se puede medir la relevancia de los resultados derivados del contexto?
¿En qué tipo de aplicaciones puede ser útil incorporar mecanismos que hagan uso del contexto para que logren robustez entregando información relevante?

### 3.6. Conclusiones

Este capítulo conforma una bisagra entre el universo real del DHD y las limitaciones de su interpretación computacional. En este sentido, se ha focalizado en determinar cuáles son las fuentes concretas de la materia prima (el ContextoDHD, sección 3.2.1) para poder producir soluciones (el Modelo Evolutivo, sección ??) a determinados efectos funcionales del DHD. De la misma manera inicia un recorrido sobre el estado del arte de propuestas referentes en las que se basa esta tesis. Entonces, desde este punto se mantendrán nuevas premisas aportadas que permitirán especializar teorías y conceptos con una nueva entidad apropiada para el DHD.

### 3.6. Conclusiones

---

---

# CAPÍTULO 4

---

## ArqDHD: Arquitectura de Software para los Dispositivos Hipermediales Dinámicos

---

### 4.1. Introducción

En este capítulo se describen los aspectos fundamentales y decisiones adoptadas en base al tipo de modelos y arquitecturas que se tuvieron en cuenta en el diseño e implementación de los Dispositivos Hipermediales Dinámicos.

Esto se logra mediante la caracterización de algunos aspectos destacados de un proceso de desarrollo específico para los DHD que se destaca por la inclusión de artefactos de arquitectura, permitiendo caracterizar de manera eficiente la semántica de aquellas componentes que juegan un papel importante y permiten resolver mas eficaz y eficientemente partes de los requerimientos funcionales anteriormente mencionado de los DHD (sección ??).

Entonces a esta caracterización la llamaremos ArqDHD con el propósito de sentar algunas precedencias hacia el estudio y definición que sirvan en los procesos de diseño de los DHD. Como ejemplo de los elementos arquitectónicos a los cuales está orientado ArgDHD se pueden mencionar:

- componentes;
- conectores;



## 4.1. Introducción

---

- contratos;
- subsistemas, o configuración de componentes y conectores;
- puertos, o puntos de interacciones con componentes;
- roles, o puntos de interacciones con conectores;
- artefactos de penetración, los cuales establecen componentes de composición o conectores internos de la arquitectura con elementos sus interfaces externas.

Conceptualmente ArqDHD encierra aquellos aspectos de la arquitectura que juegan un rol protagónico en los DHD. Esta apreciación fue motivada debido a que los requerimientos de los DHD se deben resolver a partir de la arquitectura sin tener en cuenta el modo de implementación de aquellos artefactos que la compone.

Además ArqDHD está orientada a la formalización de algunas de las propiedades arquitectónicas singulares en los DHD, promoviendo una mejor representación de la semántica para los conectores y su coordinación.

Existen múltiples definiciones de Arquitectura de Software (desde ahora en mas lo llamaremos AS), el Software Engineering Institute's (SEI)(?) recoge 75 definiciones distintas del término Arquitectura del Software. Por este motivo, se hace imprescindible elegir la definición que se ajuste más a la acepción que toma esta propuesta y especializarla en el dominio de las aplicaciones Web colaborativas.

De todas las existentes ArqDHD se centra en dos definiciones que más han influido en sus modelos:

Definición de IEEE Architecture Working Group [113]:

*La arquitectura es la organización fundamental de un sistema expresado mediante sus componentes, las relaciones entre cada uno de ellos y con su entorno, y los principios que guían su diseño y su evolución “*

Se trata de una definición consensuada por un organismo internacional, por ese motivo es un intento de simplificación y unificación de las definiciones existentes por los diferentes expertos en la materia. Lo primero que destaca de esta definición, es que establece al componente como la unidad arquitectónica utilizada a la hora de representar la arquitectura de la aplicación. El componente es un concepto demasiado ambiguo y que en ocasiones puede hacer referencia desde un subsistema hasta un componente cliente.

Además, es importante resaltar el hecho de que la definición señala a la AS como responsable de establecer los mecanismos para guiar el diseño y su evolución, ya sea mediante la captura de los requisitos no funcionales o mediante el uso de patrones. De esta manera, Booch [15] señala la importancia que tiene la AS en la

evolución de un sistema, indicando que la presencia de una arquitectura estable en un sistema, asegura las bases sobre las cuáles un sistema puede evolucionar continuamente con los mínimos desajustes y trabajo. Sin embargo, buscando una definición que fuera menos ambigua y aportara un poco más contenido a la hora de modelar el tipo de arquitectura que envuelve los DHD, ArqDHD se basa también en la siguiente:

Definición de Buschmann [17]:

*“La arquitectura es la descripción de los subsistemas y componentes de un sistema software y las relaciones entre ellos, típicamente representado mediante vistas que muestran las propiedades funcionales y no funcionales más relevantes”*

En esta definición de AS, aparecen los subsistemas junto con los componentes como unidades arquitectónicas. ArqDHD se basa en esta aproximación capturando tanto los subsistemas en el modelo de subsistemas (??), como los componentes en los modelos de integración (??) y de configuración (ver cap. 6), a la hora de representar la arquitectura Web.

Otro aspecto a destacar de la definición, es indicar que la AS debe representar los aspectos —más relevantes— del sistema, abstrayéndose de cierta información menos importante (de otra manera no se estaría representando la arquitectura, sino que se mostraría todo el sistema). Siendo así la base para proveer suficiente información para el análisis, ser ayuda en la toma de decisiones, y por lo tanto reducción de riesgos.

Hay que recordar que la arquitectura no es solamente estructural, (en ocasiones es entendida erróneamente como la estructura de la aplicación), sino que provee un mecanismo natural de integrar varias vistas del sistema. Aquí aparecen el concepto de vista y perspectiva de un sistema, que en el caso de una aplicación Web, ArqDHD ha establecido en 8 diferentes vistas. Estas vistas pueden ser tanto estructurales, funcionales y de comportamiento. Como se muestra en la sección ?? de Ingeniería Web, en ArqDHD se utilizan vistas funcionales definidas por otras aproximaciones de Ingeniería Web que son representadas de forma concurrente a la arquitectura propuesta.

En el capítulo anterior se han repasado las principales características y componentes que componen los DHD para brindar los propiedades de coordinación contratos sensibles al contexto, los cuales formarán parte de la arquitectura.

Este punto de vista permite visualizar los dominios de aplicación y los logros obtenidos en estos campos. Una vez determinado para qué sirven, el siguiente paso es describir cómo funcionan. Esta descripción se facilita si se sigue la estructura marcada por la arquitectura del sistema. Además, ...

La arquitectura del sistema está formada por cuatro subsistemas, cada uno de ellos tiene una misión específica. El diseño busca separar los conceptos, mejorando de esta manera la evolución independiente de cada subsistema 4.3.3. Para esta

## 4.2. Conceptos Fundamentales

---

representación se tuvieron en cuenta algunas de las características de los principales patrones para la representación de arquitectura, a través de los patrones de distribución definidos por diversos autores Buchmann et al. [17], Conallen [24], Trowbridge & Mancini [118]

## 4.2. Conceptos Fundamentales

Ahora se revisarán los conceptos arquitectónico mas relevante que fueron necesario tratar en las etapas de diseño.

Desde el punto de vista lingüístico, una arquitectura de software viene determinada por los componentes –elementos básicos caracterizados por una interfaz segmentada en *puertos* y *conectores* que la constituye constituyen, así como por una serie de conexiones o enlaces específicos, que definen la unión de todos ellos formando una estructura. A esta estructura se le da el nombre de configuración, y suele considerarse insertada en una jerarquía, independientemente de su granularidad. Ocasionalmente, la configuración no se describe de manera monolítica, sino que se estructura en diferentes vistas, cada una de las cuales se concentra en un aspecto diferente. Cuando lo que interesa no es obtener una configuración concreta, sino los patrones genéricos que definen a una familia de sistemas, se habla de estilos arquitectónicos [CFBS98]. En definitiva, es la definición de todos estos aspectos la que determina una visión concreta de la Arquitectura de Software; a este fin se dedicarán los apartados siguientes.

### 4.2.1. ComponenteDHD

El concepto fundamental de la Arquitectura de Software es el de componente. Esto se refiere, en términos globales, a cada una de las partes o unidades de composición –por definición– en las que se subdivide la funcionalidad de un sistema, y cuya unión da lugar al sistema completo. En su sentido más genérico, puede hacer referencia a cualquier tipo de elemento estructural, esto es, integrado en una estructura; y es precisamente con este significado con el que habitualmente se le utiliza en la Arquitectura de Software.

El término de componente no se limita a la arquitectura, sino que es de hecho utilizado en múltiples campos de la Ingeniería de Software desde la propuesta realizada por Douglas McIlroy [McI69] en la célebre conferencia de la Otan en Garmisch-Patenkirchen. En realidad, su connotación más habitual hace referencia más bien a aspectos de implementación, vinculados a los estudios de Desarrollo Basado en Componentes. Lo cierto es que, en todo caso, el término suele ir acompañado de cierta confusión: aunque existe cierto consenso en torno a una idea intuitiva, resulta muy difícil obtener una definición satisfactoria [BDH+98]. Ni siquiera los distintos intentos por proporcionar una descripción formal del concepto [Bro96] han tenido un éxito claro, de modo que el significado preciso de la noción

es aún un tanto difuso.

No obstante, la siguiente definición ha alcanzado cierta popularidad y se considera comúnmente aceptada:

Clemens Szyperski [Szy98]

*Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio.*

Clemens Szyperski [Szy98]

En el contexto del desarrollo de componentes de los DHD, se deben hacer adaptaciones, extensiones y agregados para romper de alguna manera con el alto grado de encapsulamiento - sea por abstracción, por seguridad o por motivos comerciales - con el propósito de agregarles posibilidades de adaptación mediante el uso de envolventes [Hol93], adaptadores o mediadores [GHJV94] (ver sección ??). Ese papel ha sido asumido en Arquitectura de Software, normalmente, mediante el uso de conectores desarrollados específicamente para la ocasión. Por ello, no es extraño que cada vez más autores propongan un cambio en este sentido - sobre todos en los sistemas colaborativo como el DHD -, de modo que todo componente conste de una interfaz privilegiada que permita algún tipo de acceso, con grados relativos de seguridad, a detalles internos. Esto define lo que se ha dado en llamar una caja gris [BW97], un concepto ya propuesto con anterioridad para la Orientación al Objeto [KLM+97, HL95] y que será una de las principales ideas que se adoptó en esta tesis para resolver los RequerimientoDHD (sección ??). De este modo, se obtendrá la definición de componentes como una caja gris, como un simple efecto secundario del relacionamiento con contratos sensibles al contexto.

### 4.2.2. ConectorDHD

El concepto de conector procede principalmente de los trabajos de Mary Shaw [SG96], a partir de su experiencia en Unicon [SDK+95a]. En un célebre artículo [Sha94], propuso considerar por separado las abstracciones relativas a la funcionalidad (el componente) y a su interacción (el conector). De este modo, se realiza una clara separación de intereses, que permite ampliar el nivel de abstracción y aumentar la modularidad del sistema.

Sin embargo, lo que se propone no es simplemente disponer de dos tipos de componentes, sino de distinguir dos elementos diferenciados, con funciones muy

## 4.2. Conceptos Fundamentales

---

disparos. Los componentes normales —elementos de computación— realizan una tarea sin preocuparse de cómo se relacionan con el resto del sistema; por su parte, los elementos e interacción, denominados conectores, son los que se encargan de resolver todas las cuestiones relativas a la comunicación de los primeros.

Shaw insiste en que los conectores deben ser considerados como elementos de primera clase, que tienen significado por sí mismos. Esto quiere decir que no serán definidos en función de otros elementos, ni diseñados específicamente para un componente, sino que podrán ser extraídos y considerados en otro contexto.

La forma más sencilla de ver a un conector es como la encarnación de un protocolo de comunicación, entendido en su sentido más amplio. En general, cualquier artefacto —o artefacto— que permita comunicarse a dos o más elementos es un conector. Por ejemplo, la llamada de procedimiento es un tipo clásico de conector.

No obstante, la definición del concepto supone que hay dos tipos de vínculo entre los distintos elementos de una descripción arquitectónica: por una parte, el propio conector expresa la interacción existente entre varios componentes; pero por otra, esto exige establecer, a su vez, el tipo de enlace que relaciona a cada componente con un conector determinado: este enlace recibe normalmente el nombre de adjunción o attachment. Sin embargo, a lo largo de esta tesis le denominaremos simplemente conexión, término que consideramos más intuitivo y menos forzado, y cercano a su uso convencional en español <sup>a</sup>

Es importante señalar que existen ciertas diferencias de matiz en cuanto al uso de la palabra conector. Podría decirse, incluso, que se utiliza con dos sentidos diferentes, aunque relacionados.

Por una parte, suele entenderse que la propuesta original de Shaw exige la definición de un nuevo tipo de elemento, análogo a un componente, y que se describe del modo indicado. Sin embargo, también se puede mencionar la palabra conector, de modo general, como haciendo referencia a cualquier interacción explícita entre dos componentes, lo que incluye a los *bindings* de arwin (sección ??) o a las conexiones de Rapide (sección ??). Este es, por ejemplo, el sentido en que lo usa Medvidovic en [MT00], cuando señala que la definición de conectores es uno de los rasgos que caracterizan a un Adl (sección ??). Sin embargo, en pro de la claridad, en esta tesis se utilizará la palabra únicamente en el primer sentido, es decir, como elemento de primera clase, haciendo siempre una distinción explícita entre conector y conexión.

Incluso en su forma más básica, esto es, cuando se los considera como simples conexiones, la enumeración de enlaces o asociaciones explícitas entre los elementos de un sistema proporciona una descripción, siquiera parcial, de su estructura; es decir, de su arquitectura. Puede decirse con Medvidovic [MT00], pues, que por el mero hecho de disponer de una noción de conector, un lenguaje ya está

---

<sup>a</sup>En inglés, del mismo modo, preferimos el término más general de binding, utilizado en algunos lenguajes sin conectores, como arwin, aunque es tal vez más cercano a aspectos formales o de implementación.

reforzando su capacidad para especificar configuraciones, lo que constituye la tarea principal de cualquier Adl. De hecho, uno de los primeros trabajos sobre Rapide [LVM95], uno de los lenguajes con un modelo de conexión más simple, demuestra sin embargo que esta característica, por sí sola, resulta suficiente para distinguir con claridad un Adl de un lenguaje orientado al objeto convencional. Tal vez, sugerir la posibilidad de una confusión entre estos dos tipos de lenguajes resulte ahora extraño; sin embargo, y debido al énfasis que ambos hacen en el concepto de encapsulación, éste es un punto que fue ampliamente debatido en su momento [Cle96], e incluso reaparece, todavía hoy, de manera ocasional.

Ha de tenerse en cuenta que se puede ver a los conectores de dos maneras, a menudo antagónicas, pero que no tienen por qué serlo: como una especificación, o como una simple implementación. El mejor ejemplo de lo primero son los conectores de Wright (sección ??), mientras que los de Unicon son un adecuado ejemplo de lo segundo. En un lugar intermedio podrían situarse los de C2 (sección ??). Para Wright, los conectores son ante todo especificaciones que indican qué es lo que se espera de un componente en una interacción dada. Es decir, se trata ante todo de indicar el papel de cada uno de los componentes en cada uno de los protocolos; si la especificación del conector se corresponde con las de los componentes, se puede verificar la corrección del sistema.

Para Unicon, en cambio, un conector no es más que una implementación de un protocolo; su objetivo es, ante todo, evitar que el diseñador del componente tenga que preocuparse de los aspectos de interacción. Por tanto, se plantea como un elemento reutilizable que puede ser conectado a un componente en un momento dado, y se encarga desde ese momento de realizar las interacciones apropiadas. Nikunj Mehta [MMP00] ha hecho un estudio exhaustivo de todo aquello que ha sido o es considerado un conector; aunque es el principio de una taxonomía que se hace claramente necesaria, no constituye más que un trabajo preliminar, y lamentablemente no parece haber tenido continuidad. No obstante, un trabajo de este tipo, completamente desarrollado, será imprescindible para que el concepto puede llegar a alcanzar todo su potencial.

Eventualmente, se espera que una comprensión adecuada de la naturaleza del conector permita la definición de un álgebra de conectores, que permita su combinación, permitiendo la descripción de interacciones aún más elaboradas, en las que se ha elevado el nivel de abstracción. Los conectores así obtenidos son los que han sido llamados conectores de orden superior [Gar98]; aunque en los últimos años ha habido cierto progreso en esta línea [SG01, LWF01], todavía está en sus etapas iniciales, y es aún poco más que una idea. Debe tenerse en cuenta que el propio concepto de conector está en discusión. Aunque se trata de una idea generalizada y es aceptada de manera común, no todos los autores están plenamente convencidos de su necesidad. Algunos, entre los que destaca Jeff Kramer, opinan que la existencia de los conectores distorsiona la naturaleza compositiva de una arquitectura de software, que queda afectada de manera negativa. Ciertamente, mientras que la composición de dos elementos de estructura similar –dos componentes– resulta fácil de expresar, sea formal o informalmente, es claro que la introducción de un segundo tipo de elemento –el conector– complica la situación. No se trata simplemente de una yuxtaposición de funcionalidades, sino que ha de considerarse

## 4.2. Conceptos Fundamentales

---

el tipo de composición que define el conector. Existe además toda una serie de problemas derivados, como es la diferencia precisa entre componente y conector, o cuál es la naturaleza exacta de un compuesto intermedio, en el que alguno de los extremos –o roles– de un conector quedase libre. Se ha argumentado que un “componente de interacción”, concebido de manera equivalente a un conector, pero definido de forma análoga a otros componentes, mantiene las posibilidades de reutilización sin perder la composición. Aunque esto rechaza la posibilidad que lo consideraba como una abstracción básica, no contradice explícitamente a la propuesta de Shaw: el mero hecho de plantear la diferencia entre las dimensiones de composición e interacción es ya importante, y constituye su verdadera esencia.

En definitiva, la noción de conector se ha convertido en uno de los rasgos definitorios del campo: sea como un componente específico, como una conexión compleja, o como noción por propio derecho, aparecerá siempre, de algún modo, en toda descripción arquitectónica. En esta tesis, el conector está fuertemente ligado a la noción de contrato (siguiendo los lineamientos de Meyer (70)), especializándolo para un determinado tipo de conexión, manteniendo cierto nivel de expresión y cumpliendo propiedades específicas de los DHD.

### 4.2.3. PuertoDHD

El concepto de puerto es cercano al de conector, pero no debe ser confundido con éste bajo ningún concepto. Se denomina con este nombre a cada uno de los puntos por los que un componente puede realizar cualquier tipo de interacción; dicho de otro modo, es cada uno de los fragmentos en los que se segmenta el interfaz de un componente.

En definitiva, hace referencia a un punto de entrada o de salida de la caja negra que es, de hecho, el componente. Distintos autores lo ven y denominan de distinto modo, y la analogía que implícitamente se establece no es idéntica en todos los casos.

Los puertos se han denominado de distintos modos en distintos lenguajes; así, en *Wright* se llaman puertos en los componentes, pero roles en los conectores, mientras que en *Unicon* reciben el nombre de jugadores. En *arwin* se conocían inicialmente como puertos, para después pasar a llamarse nombres de interfaz, de manera consistente con su semántica en cálculo-.

Actualmente conocen como portales, un nombre que expresa total neutralidad respecto del signo de la interacción: es decir, pueden ser de entrada o de salida.

Habitualmente los puertos se agrupan definiendo una interfaz; en algunos lenguajes se permite, incluso, que definan más de una. Incluso en algunos sistemas se asume que el puerto está subestructurado en varios puntos de entrada, y por tanto se define todo él como una interfaz completa. Ése es el caso, por ejemplo, de Koala [vOvdLKM00].

ArqDHD al estar fuertemente orientado a los conectores, en particular la familia de conectoresDHD (sección ??) trata conceptualmente a los puertos como partes de los conectores, esto puede verse reflejado en las referencias en cuanto al diseño. Entonces, en estos casos no conforma un aspecto fundamental, ya que no configura la naturaleza externa de los componentes, lo que a su vez no condiciona la estructura de la arquitectura.

## 4.3. Visión arquitectónica de los DHD

Ahora se inicia a definir lineamientos que permitan describir algunas de las características que se pretenden estudiar sobre ArqDHD. Se comienza con el recorrido de diferentes propuestas que involucran determinados aspectos de ArgDHD y sus interconexiones conforman su espacio de estudio.

### 4.3.1. Propuestas de Arquitectura del Software para los DHD

A diferencia de lo que ocurre en la parte funcional de la aplicación Web, es difícil establecer criterios y formalismos en cuanto a qué aspectos (incluyendo los conceptuales) es necesario capturar en la arquitectura Web colaborativa. Por lo tanto, es importante mostrar los diferentes tipos de aproximaciones existentes, cuáles son sus principales características comunes y qué aspectos comparten con ArqDHD. De esta manera comienzan a identificarse las características que distinguen a ArqDHD del resto de aproximaciones que representan la arquitectura de las aplicaciones Web.

- REST

El Representational State Transfer (REST) [31] consiste en un estilo arquitectónico para aplicaciones Web distribuidas. REST está basado en la definición de un conjunto de restricciones arquitectónicas centradas en el dominio de las aplicaciones Web.

REST propone un proceso de definición de su estilo arquitectónico apoyándose en la introducción de restricciones sobre otros estilos arquitectónicos reconocidos dentro de la arquitectura del software como Client/Server, Layered, Cache, etc. REST hace uso de tres vistas arquitectónicas (Proceso, Conector y Datos) para especificar la arquitectura Web. En su especificación ignora los detalles de la implementación del componente y la sintaxis del protocolo para enfocarse en los roles de los componentes, las restricciones de las interacciones con otros componentes y su interpretación de los elementos de datos significativos. Las restricciones de REST se fundamentan en los componentes, conectores y los datos que definen la base de la arquitectura Web.

El objetivo de REST es ser una guía para que los diseñadores de aplicaciones Web definan el conjunto de requisitos no funcionales relevantes para la apli-



cación. Ejemplos de este tipo de requisitos son la obtención de una buena escalabilidad, despliegue independiente, reducción de la latencia de interacción entre los componentes, refuerzo de la seguridad o la encapsulación de los sistemas legados.

La sintaxis utilizada para modelar la arquitectura no se basa en el uso de estándares de modelado, ni propone su formalización mediante metamodelos. Esto hace difícil su especificación mediante herramientas comerciales.

La clasificación de tipos de componentes realizada por el estilo arquitectónico REST ha servido de referencia para la definición de la tipología de componentes definida por los estilos de ArqDHD.

- **Architecture Recovery of Web Applications**

Hassan y Holt [42] consideran que las aplicaciones Web se convertirán en los próximos años en sistemas legados de difícil mantenimiento. Partiendo de esta base, proponen una aproximación que realice una recuperación de la arquitectura de las aplicaciones Web a partir de su implementación. El objetivo de esta ingeniería inversa de la arquitectura es mejorar el mantenimiento de la aplicación, al hacerla más entendible para los desarrolladores.

La aproximación define un conjunto especializado de parsers/extractores que analizan el código fuente y binario de las aplicaciones Web existentes y obtienen un conjunto de diagramas de arquitectura situados a diferentes niveles de abstracción.

La aproximación establece la definición de un conjunto de cinco componentes identificados como comunes dentro de la arquitectura Web: Static Pages (Paginas Estáticas), Active Pages (Paginas activas de servidor), Web Objects (componentes de servidor), Multimedia Objects (objetos multimedia como imágenes, video y sonido) y Databases (bases de datos).

La representación de la arquitectura se basa en un conjunto de tres modelos que de forma progresiva reducen los detalles obtenidos en la recuperación de datos, para mostrar únicamente la información relevante para la arquitectura. El primer modelo es ELS (Entity-Level Schema). Este modelo es el nivel de abstracción más bajo y muestra las relaciones entre los elementos que viven dentro de los componentes Web (objetos, tablas, variables, etc.). A partir de este modelo se definen las transformaciones necesarias para subir el nivel de abstracción hasta el modelo CLS (Component-Level Schema). CLS representa las relaciones entre los componentes de la aplicación Web (StaticPages, ActivePages, Databases, etc.). El último nivel de abstracción es el modelo ALS (Abstract-Level Schema) que representa las relaciones entre los elementos de mayor granularidad, los subsistemas y los componentes que contienen.

Los diferentes modelos representados en esta aproximación se basan en esquemas (Schemas) que son básicamente modelos Entidad-Relación (EER).

- **WAE: Web Application Extension of UML**

La propuesta WAE [24] (Extensión para las aplicaciones Web) definida por Jim Conallen propone un conjunto de modelos UML cercanos a la implementación que, dentro del contexto del Proceso Unificado de Rational (Rational Unified Process [47]), giran en torno a la arquitectura de la aplicación. Conallen basa la descripción de la arquitectura de las aplicaciones Web en el

conocido trabajo de Kruchten —The 4+1 View Model of Architecture“ [64], estableciendo los artefactos utilizados en cada una de las vistas que define Kruchten para el desarrollo de las aplicaciones Web.

Conallen, partiendo de la idea que una arquitectura nunca aparece de la nada, se basa en un conjunto de patrones de arquitectura para su definición en la fase de diseño. Por un lado, adapta patrones comunes que considera particularmente adecuados para las aplicaciones Web, como son: Façade de Gamma et al. [36], Page Composition y Template Page. Otros patrones específicos para la capa de presentación son Thin Web, Thick Web y Web Delivery.

Sin embargo, donde verdaderamente la arquitectura comienza a tomar relevancia es dentro del proceso definido por Conallen en la fase de diseño. En esta fase define WAE, que incluye un conjunto de tipos de componentes especializados en el dominio de las aplicaciones Web (p.e. Server Page, Client Page, HTML Form, etc.). La definición de cada uno de los componentes la realiza mediante el mecanismo de perfiles proporcionado por UML. WAE permite representar una aplicación Web muy detalladamente, acercándose al nivel de implementación concreta, con la introducción de aspectos dependientes de ASP y JSP. A partir de la representación en WAE, existe un mecanismo de generación automática que permite obtener el esqueleto de los diferentes componentes.

- WebArquitect

WebArchitect [112] es otra propuesta que se centra en la arquitectura y las funciones de los lugares Web, más que en la apariencia de cada página. Este método comienza con una actividad de análisis en la que, mediante el modelo Entidad-Relación, se representa el dominio del problema. A continuación, un análisis de escenarios determina cómo los usuarios potenciales interactúan con la aplicación Web para cumplir los objetivos de negocio. A partir de las fases de análisis, la arquitectura de la aplicación es diseñada en la fase de diseño arquitectónico. La arquitectura es representada mediante un modelo denominado RMDMW (Relationship Management Data Model for Web-Based Information Systems) que consiste en una extensión del modelo propuesto por RMM [48], con la introducción de eventos, roles y productos.

Mediante RMDMW el diseñador determina la navegación y el modo de mapear<sup>1</sup> la navegación a las diferentes páginas.

El método también define atributos para cada página, que son utilizados para el mantenimiento de la misma. La implementación y mantenimiento de la aplicación resultante es soportada por una herramienta del mismo nombre, que permite a los diseñadores manipular de forma directa metaenlaces entre páginas organizadas en un árbol jerárquico. Por otro lado, la visualización de las aplicaciones resultantes se realiza mediante un cliente Web denominado Pilot-Boat, que navega y deja que los usuarios colaboren a través de los lugares Web.

- WAM (WebComposition Architecture Model)

WAM [70] es una aproximación muy reciente basada en la extensión de la reconocida aproximación WebComposition [35]. WAM introduce una des-

cripción arquitectónica que sirve como mapa para el mantenimiento de las trazas de las interrelaciones entre diferentes aplicaciones Web federadas. Las aplicaciones Web federadas se basan en la idea de las aplicaciones Web que comparten componentes y elementos realizados por múltiples proveedores a lo largo de la Web. Entre los artefactos modelados están los servicios Web, las propias aplicaciones Web y las zonas organizacionales de control que están sujetas a la evolución en el sentido de la aproximación WebComposition.

WAM persigue hacer comprensible mediante los modelos la estructura técnica de la federación a los actores (stakeholders) que intervienen en el desarrollo de la aplicación Web (p.e. arquitectos, desarrolladores y administradores).

WAM se basa en DSLs (Domain-Specific Languages) para representar los diferentes elementos identificados como relevantes en la arquitectura Web. Estos son: service, application, data provider, process unit, invocation y trust relationship. A partir de los elementos del modelo, se establece un mapeo a un lenguaje llamado WAM-XML que sirve de base para su tratamiento en herramientas y para dar soporte a los sistemas.

- OOHDM -Java 2

OOHDM-Java 2 [51] es un trabajo que propone una línea de producto en J2EE para simplificar el desarrollo de aplicaciones utilizando la conocida aproximación de Ingeniería Web OOHDM [106]. OOHDM-Java2 es una arquitectura que permite desacoplar las decisiones de diseño relacionadas con el modelo de dominio de aquellas relacionadas con la navegación y la arquitectura de interfaz. OOHDM-Java 2 tiene asociado un framework J2EE que extiende el concepto del patrón de Buchmann et al. [17] Model-View-Controller y realiza una separación de los nodos de navegación de sus interfaces. Así, introduce la idea de objeto de navegación, y reconoce el hecho de que la navegación puede ser dependiente del contexto. La estructura de OOHDM-Java 2 contiene elementos que configuran una arquitectura que persigue las mejores prácticas y mejores resultados de mantenimiento. Este esqueleto (framework) es instanciado para las diferentes aplicaciones definidas en OOHDM, mediante un conjunto de tareas definidas que debe seguir el diseñador para su utilización.

#### 4.3.2. Propuestas basadas en el Desarrollo Dirigido por Modelos para Aplicaciones Web

El último conjunto de propuestas de interés para este trabajo es aquel que se basa en el paradigma de ingeniería dirigida por modelos (MDE) [55] para el desarrollo de las aplicaciones Web. MDE proporciona como principales ventajas el dotar de un mecanismo de trazabilidad desde los modelos hasta la implementación mediante el uso de las transformaciones. Sin embargo, debido a la juventud de MDE son pocas las herramientas disponibles hasta el momento y la mayoría de las implementaciones realizadas por las aproximaciones no se basan en transformaciones formales, sino que las implementan directamente mediante generación de código. Otro de los aspectos a considerar es el uso de los modelos definidos en la

Ingeniería Web para representar las diferentes vistas: mientras que algunas aproximaciones definen sus propios modelos, otras se valen de trabajos reconocidos para definir sus modelos

- **Model-Driven Development of Large-Scale Web Applications**

Tai et al. [111] definen una aproximación basada en MDA para el desarrollo de aplicaciones Web de gran tamaño. Para ello, proveen un conjunto de modelos basados en un metamodelo que es usado como contrato principal entre los desarrolladores.

El metamodelo juega un papel fundamental al realizar una división en subaplicaciones que provee la especificación y que todos los artefactos deben cumplir. El metamodelo se ha definido conforme a la arquitectura propuesta para la plataforma J2EE. En un futuro se pretende hacer que el metamodelo sea general para otras plataformas.

Los modelos definidos por la aproximación son: (1) el modelo de transición de páginas (navegación entre los nodos mediante un gráfico), (2) modelo de flujo de página (composición de la interfaz de una página) y el (3) modelo de datos (información que se almacena a partir de las páginas). Para la definición de estos modelos, Tai no se ha basado en los modelos definidos dentro de la Ingeniería del Software.

Acompañando a esta aproximación se ha definido una herramienta llamada WAST (Web Application development Support Tool). WAST provee un conjunto variado de generadores de código (básicamente esqueletos) y mecanismos de validación de código para comprobar que los artefactos definidos en las diferentes vistas (p.e. los JSPs (Java Server Pages)) son compatibles con el modelo. Como parte del proceso de desarrollo se deben definir los diferentes actores (stakeholders), cada uno de los cuáles debe seguir las reglas impuestas por el metamodelo. Estos actores son el diseñador de pantallas, el mantenedor de modelos, el programador de lógica de negocio, el programador de objetos de servidor y el ensamblador de la aplicación. Los programadores deben rellenar aquellos huecos que los generadores de código dejan, siempre verificando que el código introducido respecta el modelo.

- **MIDAS**

MIDAS [18] metodología dirigida por modelos para el desarrollo de aplicaciones Web. Esta metodología aplica un metamodelo MDA a la plataforma Web utilizando XML y tecnología objeto-relacional. MIDAS propone diferentes modelos PIM y PSM y define algunas reglas de mapeo entre los modelos.

Los modelos independientes de plataforma (PIM) propuestos por MIDAS están definidos utilizando el estándar UML [120]. Los modelos PIM están constituidos por contenido, navegación y presentación. Para representar los distintos modelos MIDAS se basan principalmente en UWE [61].

MIDAS también define un conjunto de modelos dependientes de plataforma (PSM) que representan cada una de las vistas definidas como PIM. Así, para representar el modelo de contenido dependiente de plataforma, se ha valido de la tecnología objeto-relacional. Sin embargo, para representar la navegación y la presentación ha utilizado XML.

MIDAS propone únicamente las guías para realizar las transformaciones PIM- PIM, PIM-PSM y PSM-PSM necesarias para completar su desarrollo. Actualmente, está trabajando en implementar las transformaciones para obtener la aplicación final sobre plataformas como J2EE y .NET.

- Model-Driven Development Process for UWE

El reciente trabajo [58] redefine el proceso de UWE estableciéndose como un proceso de desarrollo dirigido por modelos (ahora llamaremos MDD-UWE). Para su definición se ha basado en los principios de MDA, usando los estándares OMG para la definición de sus modelos y transformaciones. El proceso de desarrollo de MDD- UWE consiste en un conjunto de modelos y transformaciones cuya especificación está soportada por metamodelos y reglas de transformación. Los metamodelos son el metamodelo de MDD-UWE [59], el metamodelo Web Requirements Engineering metamodel (WebRE) [30] y además, utiliza el metamodelo de nuestra propia propuesta ArqDHD (Meliá & Gómez [75]).

El proceso de desarrollo especifica los requisitos funcionales basándose en el trabajo de [29] para su automatización de requisitos a contenido funcional. Los aspectos funcionales son especificados simultáneamente con los modelos de arquitectura especificados por ArqDHD, y posteriormente son integrados. Existe otra alternativa para la integración de la arquitectura con un modelo —Big Picture“, es decir, un modelo que integra ya las tres vistas funcionales (presentación, contenido y navegación).

Las transformaciones definidas se basan en tecnologías y lenguajes diferentes, desde el lenguaje estándar de transformaciones QVT [91], ATL [5], transformaciones basadas en grafos, e incluso transformaciones implementadas en código java en la propia herramienta ArgoUWE [3].

- Consistent and Adaptable W2000 Models

Esta propuesta ha evolucionado a través de un reciente trabajo [9] que establece el metamodelado de todos los modelos de W2000 mediante MOF. Además, propone la definición de las reglas de transformación que permitan a los usuarios acceder y controlar la consistencia de los artefactos producidos por el proceso W2000 y posteriormente adaptarlos en un modo controlado.

La propuesta no propone un proceso de desarrollo determinado ya que prefiere dotar de libertad a los desarrolladores para que elijan el que prefieran. La corrección de los modelos definidos por los modeladores es controlada mediante la definición de los modelos como instancias de metamodelos MOF y mediante el establecimiento de las restricciones oportunas mediante el estándar OCL [93].

Por otro lado, se propone la definición de reglas de transformación definidas mediante el lenguaje de grafos AGG (Attributed Graph Grammar System) [28] que representan las reglas de transformación a nivel de metamodelo.

Además, la aproximación tiene el soporte de una herramienta realizada mediante un plug-in2 de Eclipse [27] para su parte gráfica, un repositorio MOF comercial, MDR [79], que le permite la carga de los modelos mediante XMI y la herramienta proporcionada para el lenguaje AGG [28] para definir las reglas de transformación

### 4.3.3. Situación de ArqDHD en la Investigación Actual

Realizando un ejercicio de revisión conjunta de las tres tendencias consideradas influyentes para el presente trabajo, es muy interesante destacar cuál ha sido la evolución en cada una de ellas y en qué elementos se tuvieron en cuenta en nuestra aproximación de modelo. La Figura ?? muestra una representación del espacio basada en tres coordenadas, cada una de las cuáles representa una tendencia: Ingeniería Web, Arquitectura Web y MDE. Situándose en un estado evolutivo dentro de cada coordenada o tendencia, ArqDHD constituye un cubo cuyo volumen representa el espacio que ocupa dentro de la investigación actual.

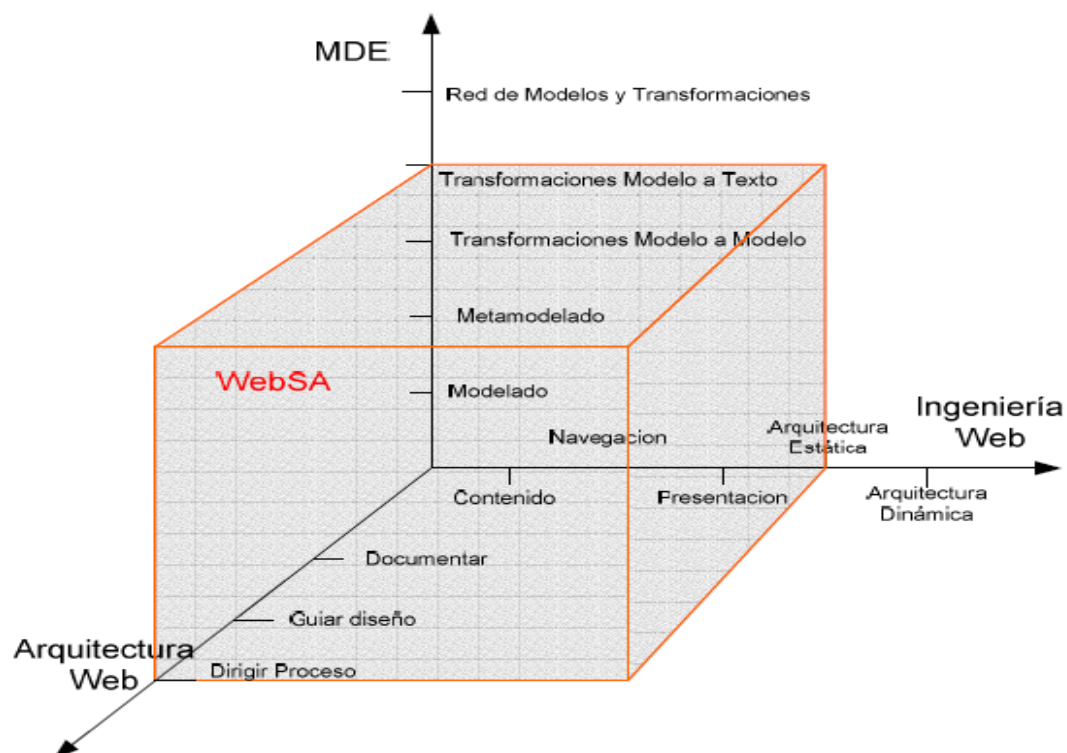
La coordenada X está representada por la Ingeniería Web, y en ella se presentan las diferentes vistas que se han ido introduciendo a lo largo de su existencia. En ella puede apreciarse cómo las tres vistas funcionales son las primeras en ser capturadas: contenido, navegación y presentación. La cuarta vista, también capturada por ArqDHD, es la arquitectura estática. La única vista que no es representada por ArqDHD es la arquitectura dinámica, que representaría el comportamiento tanto externo como interno de los componentes. Esta tarea no ha sido cubierta por el presente trabajo.

La coordenada Y es ocupada por la ingeniería dirigida por modelos (MDE) y en ella se muestran los diferentes estadios por los que ha ido pasando: representación de las vistas mediante modelos (modelado), formalización de los modelos mediante el metamodelado, definición de las transformaciones modelo a modelo para establecer la traza entre los diferentes modelos del proceso y, por último, las transformaciones modelo a texto que permiten obtener la implementación a partir de los modelos.

ArqDHD ha completado las diferentes tareas de MDE. Queda sin embargo un largo camino de interoperabilidad entre los modelos y transformaciones realizadas, que permitan establecer una red para compartir los recursos definidos en MDE.

Por último, la coordenada Z la representa la Arquitectura Web. En ella se muestra una serie de puntos que representan el papel que ha tenido la arquitectura dentro de las aproximaciones. En un principio la arquitectura únicamente se utilizaba para documentar. Poco a poco se ha ido extendiendo su uso como guía para definir el diseño. El último estadio, que es el ocupado por ArqDHD, es aquel en el que la arquitectura define los artefactos más importantes, dirigiendo el proceso de desarrollo

### 4.3. Visión arquitectónica de los DHD



#### Estilo Arquitectónico

Existe una gran variedad de opiniones al respecto al significado de estilo arquitectónico. Una concisa definición proporcionada por Shaw & Clements [109], define un estilo arquitectónico como *“un conjunto de reglas de diseño que identifica los tipos de componentes y conectores que pueden ser usados para componer un sistema junto con las restricciones en el modo en que la composición es hecha”*. Una similar posición es expresada por Medvidovic [69]. De manera que un estilo es entendido como un vocabulario para expresar arquitecturas. Basándonos en estas definiciones queda claro, que se establece un vínculo entre el estilo arquitectónico y la unidad arquitectónica.

En ArqDHD se establece en la fase de análisis dos estilos arquitectónicos que representan el sistema basándose en diferentes unidades arquitectónicas y relaciones.

Por un lado, el modelo de subsistemas sigue el estilo de capas Buchmann et al. [17], donde es el subsistema la unidad arquitectónica, y las relaciones de dependencia son los conectores. De forma paralela se define el modelo de configuración que sigue un estilo arquitectónico, donde es el componente Web la unidad arquitectónica, y en este caso son los conectores mediante puertos e interfaces el pegamento entre cada uno de los componentes.



En este caso, la forma de representación de un patrón se realiza mediante una determinada configuración de componentes que se aplica para resolver una cuestión de integración entre diferentes subsistema que tienen comportamientos aislados. A continuación se enumeran los patrones representados por diferentes autores y son separados en función de las capas en las que son aplicados:

- **Model-View-Controller (Modelo-Vista-Controlador)** (Buchmann et al. [17]): patrón que permite separar el modelado del dominio, la presentación y las acciones basadas en las interacciones del usuario en 3 componentes principales. (1) Modelo: que maneja el comportamiento y los datos del dominio de la aplicación, respondiendo a las peticiones de información sobre su estado (usualmente desde la vista) (2) Vista: muestra la información al usuario final, (3) Controlador: recibe las peticiones del usuario, informando al modelo y/o la vista para cambiar. Existen diferentes variantes que pueden introducir cambios de comportamiento
- **Page Template (Página Plantilla)** (Conallen [24]): Este patrón define una única página plantilla que genera todas las páginas Web que obtiene el cliente. Su característica especial es que la página plantilla referencia a cada una de las partes o fragmentos de página que contiene dinámicamente. Además, la configuración de la plantilla puede estar almacenada en un fichero independiente, lo que permite gestionar la apariencia sin recompilar el código.
- **Page Controller**(Trowbridge & Mancini [118]): Es una especialización del MVC, que establece un controlador común llamado BaseController que contiene todos los componentes o partes de la página Web que son comunes al resto de controladores de cada una de las páginas. Consigue reutilizar aquellos componentes o acciones que se van a repetir en múltiples controladores de página.
- **Front Controller**(Trowbridge & Mancini [118]): Otra versión del MVC, propone que se establezca un único controlador que centralice todas las peticiones, y que esté separado en dos aspectos: un manejador de las peticiones del cliente y disparador de las peticiones. Este patrón de arquitectura hace uso del patrón de diseño Command (Gamma et al.[36]) para gestionar las peticiones que el controlador recibe de la interfaz. Por otro lado, debido a que el estilo arquitectónico permite modelar estos patrones, esto proporciona al arquitecto introducir pequeñas variantes a cada uno de ellos, sin perder la finalidad última del patrón. Por ejemplo, la evolución del MVC en MVC2, donde se independiza la navegación de la vista, mejorando la independencia de la vista con el controlador.

#### 4.3.4. El contrato como conector

Entre la diversidad de propuestas de diseño de sistemas e-learning con características adaptativas, existen diferentes variantes metodológicas y tecnológicas, una de ellas es la incorporación de contratos context-aware. En el capítulo 5,



### 4.3. Visión arquitectónica de los DHD

---

expusimos las diferencias más significativas entre los sistemas tradicionales para e-learning y nuestra actual propuesta sobre los dispositivos hipermediales context aware dinámicos para educación investigación.

Como avance de la Tercera Fase de las tesis doctoral en curso Contratos para Context-Aware dinámico

b

y mostrando resultados de las fases anteriores, en este capítulo explicitaremos, qué es un contrato context-aware, cuál es su complejidad y, dónde y cómo se lo puede aplicar.

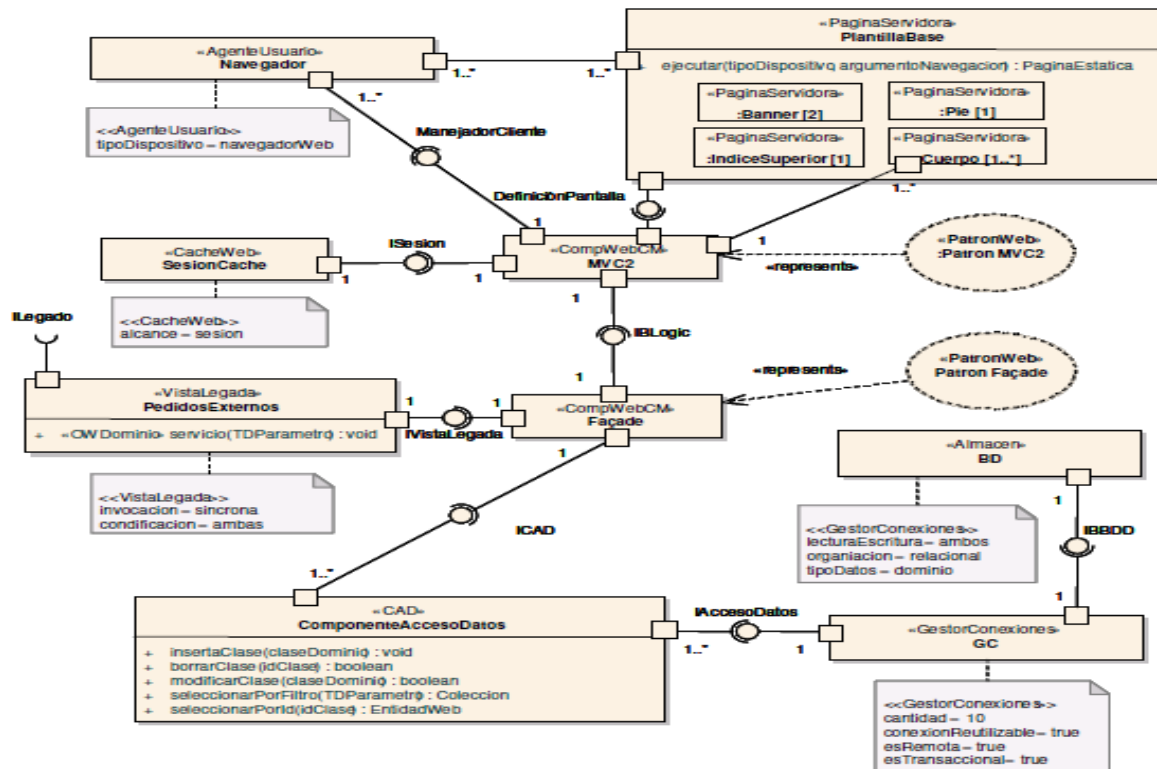
Conceptualizar al contrato como una componente de primera clase, donde las demás componentes que lo relacionan dependen de su funcionamiento, nos brindará una visión más completa orientada a la arquitectura y a la conexión entres sus componentes. Pensar el contrato como una pieza de conexión (referenciado con el termino “connectors”, conectores, en el área de arquitecturas en ingeniería de software) nos permitirá incorporar (conectar) nuevos modelos independientes a los llamados framework elearning en términos de dicha ingeniería.

En la figura ?? representamos la integración de un modelo externo con unfamework e-learning por medio de un conector referenciado con el nombre componente contrato.

---

<sup>b</sup>El presente capítulo y desarrollos, gráficos y tablas sobre Context aware del capítulo anterior, forman parte del texto de la Tesis Doctoral en desarrollo del Becario Lic. Alejandro Sartorio. Doctorado en Ciencias Informática de la Facultad de Informática de la Universidad Nacional de La Plata. Plan de trabajo aprobado por resolución HCA 3300-6865/000 del 16-12-05 (Univ. Nac. de la Plata) Director: Dr. Gustavo Rossi, Codirectora: Dra. Patricia San Martín

### 4.3. Visión arquitectónica de los DHD



Representación del contrato como conector entre un framework colaborativo (tipo e-learning) y un modelo externo.

La componente modelo externo representa una entidad, puede ser un modelo conceptual, una herramienta, applets, API (Application Programming Interface), o cualquier sistema independiente que cumpla con la funcionalidad de brindar información de contexto para ser incorporada en la semántica de la componente contrato. Las herramientas y servicios que componen el framework e-learning son los puntos de comunicación con la componente contrato. Luego, la entidad sistema representa el ambiente del servidor donde se encuentra el entorno de la plataforma elearning.

Este entorno abarca servidores web, servidores de base de datos, sistemas operativos, archivos y repositorios de recursos, sistemas institucionales, etc. La mayoría de los clientes deben ser estándares (similares a Web Browsers) y, las salidas de las aplicaciones deben poder ser presentadas a los clientes usando lenguaje de marcas tipo HTML.

En el próximo punto, nos centraremos en la importancia del tipo de modelado y diseño con los que podríamos tratar a los contratos.

#### Perspectiva de modelado

El modelado y diseño de sistemas orientados a servicios complejos, no sólo contribuye a una mejor comprensión de los problemas o posibles soluciones que se pueden adoptar en un proyecto de I&D como Obra Abierta, sino que facilita la comunicación entre el grupo de investigación, especialmente cuando se encuentran físicamente separados y/o afiliados a diferentes organizaciones.

El trabajo de modelado, es la base de procesos de desarrollo orientados a servicios para educación y/o investigación, que debe guiar tanto a los diseñadores como a los desarrolladores en el relevamiento de los requerimientos pedagógicos y/o investigativos para implementarlos en un artefacto – objeto- de software. Esta tarea de modelado puede brindar una guía clara de trabajo al equipo de I&D, entregando las soluciones a tiempo y facilitando el logro de un producto final de alta calidad, con bajos riesgos en el desarrollo.

Atendiendo a los cambios de la tecnología y a la continua evolución de los estándares, el correcto modelado y diseño de soluciones Web complejas, se torna cada vez más importante. Resultado de esto, es el nuevo paradigma de desarrollo en sistemas llamado “Model-Driver Architecture” (MDA).

MDA está focalizado en la importancia de los modelos de plataformas independientes y plataformas específicas, que permiten una separación de la abstracción del dominio de conocimiento del particular entorno de implementación. En la actualidad, dependiendo de los objetivos de las plataformas, con el desarrollo de herramientas avanzadas para el diseño a partir de códigos de programación, el rol que cumple el modelo (o modelado) comienza a ser fundamental. A través del uso de MDA, el modelo representa a la programación de alto nivel construida para desarrollos de aplicaciones como por ejemplo, un dispositivo hipermedial dinámico como el de Obra Abierta, con requerimientos factibles de trazar (en el sentido del modelado desde el punto de vista de la Ingeniería de Software). Las herramientas avanzadas pueden proveer generación automática de código XML, Java u otro código de programación a partir de un determinado modelo. Un correcto método de modelado debe brindar el soporte necesario para poder tomar la decisión sobre qué parte de la aplicación debe ser expuesta como servicio, o qué parte de la arquitectura debe ser confeccionada para invocar a los servicios Web.

El modelado de soluciones orientadas al servicio no es una tarea simple. Los servicios heredan algunas características de sus predecesores – objetos y componentes y también usan elementos del flujo de tareas y de los procesos de negociación. De esta manera, el modelado de servicios debe cubrir totalmente las diferentes facetas funcionales y tecnológicas para su prestación e implementación. El uso de Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) como notación semi-formal para el modelado de servicios es en la actualidad, una elección consensuada por la comunidad informática.

Aunque en su origen, UML fue concebido para el modelado de sistemas orientados a objetos, puede ser fácilmente extendido para soportar modelos de otros

conceptos tal como actividades de negocios, interfaces de usuarios y esquema de datos. Sin embargo, podemos observar que para el modelado orientado al servicio, todavía este lenguaje se encuentra en una etapa inicial, ya que no ofrece mecanismos para la representación de componentes y servicios a nivel lógico.

En Obra Abierta adoptamos estrategias y modelos basados en el concepto de componentes de servicios; similar a la propuesta de Zoran (2006), en su publicación *Modeling Services and Components in a Service-Oriented Architecture*. Una componente servicio es un bloque principal que interviene en algún tipo de relación entre objetos del sistema, las componentes que conforman a los servicios para educación y/o investigación son modeladas como contratos de servicios en donde se efectúan algunos de los procedimientos pedagógicos o investigativos (en forma de reglas) como ejecuciones de operaciones entre los objetos.

El concepto de la componente contrato ha sido introducido en el capítulo anterior, como una pieza de software que permitía la adaptación de servicios al contexto de los objetos que lo utilizaban, o sea como componente para establecer algún tipo de relación “controlada” entre ellos. En este capítulo, a los contratos los abordaremos desde la perspectiva de los MDA, haciendo referencia al propio modelo adoptado en Obra Abierta.

#### 4.3.5. Los conectores en entornos adaptativos

##### Aspecto Dinámicos de la Arquitectura DHD

Dichos sistemas son altamente dinámicos, y muestran una estructura evolutiva. Continuamente se añaden, eliminan, o reemplazan componentes, a veces incluso como parte del funcionamiento normal. Estos detalles ya no son coyunturales, sino estructurales, y deben tenerse en cuenta durante el Diseño del sistema. Por tanto, para ser realmente útil, un Lenguaje de Descripción de Arquitectura (Adl) debe ser capaz de especificarlos. Sin embargo, esto no es lo normal en los Adls existentes, cuya naturaleza es mayoritariamente estática. Los pocos que muestran capacidades dinámicas muestran también limitaciones, que dependen en gran medida de sus orígenes particulares.

Cuando la evolución de una arquitectura es continua, y los cambios en su interior siguen un patrón predefinido, entonces ha de considerarse que este patrón forma parte intrínseca de la estructura. El objetivo de la Arquitectura de Software es describir la estructura de los sistemas, y esto debería incluir tanto a sus partes fijas –estáticas–, como a las cambiantes –dinámicas–. De hecho, en muchos sistemas, el rasgo diferencial, que lo distingue de otros sistemas similares, es el comportamiento dinámico de su arquitectura; a menudo, puede resultar aún más importante que el esquema estático de partida. Algunos ejemplos de sistemas para los que la definición de dinamismo es esencial son los siguientes:

- Los sistemas abiertos y distribuidos, cuya estructura cambia de manera cons-

### 4.3. Visión arquitectónica de los DHD

---

tante, a medida

- Los sistemas evolutivos de varias clases, que se ajustan a un cierto patrón.
- Los sistemas multi-agente (Mas), dentro y fuera del contexto de la Inteligencia Artificial; y en general todos los sistemas capaces de expresar algún grado de movilidad.
- O, por ejemplo, los sistemas reflexivos, que son interesantes en sí mismos, pero que además constituyen la base de la propuesta final de este trabajo.
- Los sistemas continuos, de tiempo real y tolerantes a fallos, en los que a menudo se requieren cambios, pero que no pueden ser detenidos para ser actualizados.
- Los sistemas adaptativos o auto-organizados, que modifican su propia estructura para adaptarla a cada entorno concreto.

El DHD es parte de la última familia de sistemas enumerados, donde se permite a través de una característica arquitectónica poder resolver requerimientos funcionales, en este caso de adaptación, mediante la conjunción (conexión) coordinada de subsistemas.

Hoy en día, todas estas propuestas han sido desarrolladas, y muchas otras se han ido incorporando; muchos sistemas anteriores han sido adaptados o reconocidos como Adls dinámicos, y se han desarrollado varias tesis doctorales en todo el mundo [Cra97, MG99, Med99, dP99, Sch99, Wer99, Pry00, CV00] dedicadas a este tema de algún modo. En definitiva, el desarrollo e interés creciente de este campo es innegable, y se presenta, sin duda alguna, como uno de las líneas de trabajo más fructíferas de la próxima década.

#### 4.3.6. Tipos de Dinamismo

Todo sistema software cuya estructura es descrita como una arquitectura tiene siempre asociado algún tipo de comportamiento dinámico. Ahora bien, es obvio que este dinamismo no siempre indica el mismo grado de variabilidad. Algunos sistemas muestran una estructura inalterable, en la que los distintos elementos se limitan a comunicarse; en otros, la topología interna se altera continuamente, e incluso se incorporan nuevos tipos de elementos.

Resulta fundamental, para poder realizar una comparación adecuada entre las distintas aproximaciones, delimitar los diferentes tipos de dinamismo que pueden aparecer en un sistema software.

La clasificación que se incluye más abajo ha sido planteada con este único objetivo. No es exacta, ni pretende ser determinante; el límite entre las distintas categorías puede aparecer difuminado en algunos casos. Sin embargo, parece servir adecuadamente a su objetivo, esto es, aclarar el tipo de actividad al que se hace

referencia en cada caso; y ya ha sido publicada en dos ocasiones [CFBSB00, CFBSB01].

Según esto, entonces, pueden identificarse los siguientes tres tipos de dinamismo, en orden creciente de potencial para el cambio:

- **Dinamismo Interactivo.** Este término se refiere al movimiento de datos entre componentes, el dinamismo inherente en la comunicación e interacción usual. Se relaciona con cuestiones tales como la sincronización y los protocolos, y afecta, no a la estructura, sino al comportamiento de un sistema computacional. Está presente de manera implícita en todo sistema, pero resulta particularmente obvio en los concurrentes y paralelos.
- **Dinamismo Estructural.** Hace referencia a los cambios que afectan a los propios componentes y sus interacciones, y que normalmente se expresan como la creación y destrucción de instancias y enlaces entre componentes. Por tanto, la arquitectura de software del sistema resulta alterada, porque lo que cambia es la estructura. Este tipo de dinamismo es natural en sistemas distribuidos.
- **Dinamismo Arquitectónico.** Se refiere a los cambios en los tipos de componente e interacción, modificando por lo tanto incluso el significado y “vocabulario” del sistema. Se expresa como la inserción de nuevos tipos de componente (y conector), y el borrado de algunos ya existentes. Ahora, lo que cambia es la infraestructura, el sustrato en el que las arquitecturas se definen, su meta-nivel. Este es el nivel definitivo de dinamismo, y también el más difícil de capturar y utilizar. Aún hay que demostrar su verdadera importancia y utilidad, pero está claramente presente en los (auténticos) sistemas abiertos.

#### 4.3.7. Reconfiguración Dinámica

Hay al menos dos aproximaciones a la especificación de dinamismo en sistemas software. Una de ellas trata de encontrar un conjunto adecuado de comandos imperativos, capaces de expresar cualquier configuración deseable. La otra utiliza una sintaxis declarativa, que es generalmente más potente e intuitiva, pero no siempre tiene la flexibilidad deseada. El trabajo sobre este tema se centró inicialmente en el primer enfoque; sin embargo, a medida que se alcanza una mejor comprensión del dinamismo estructural, la presencia del segundo tiende a aumentar.

A menudo, la denominación de Arquitectura de Software Dinámica ha sido discutida. Muchos autores prefieren el término Reconfiguración Dinámica, que ha sido consagrado por años de trabajo previo en el campo específico de la configuración de sistemas distribuidos. Con esa expresión se hace referencia, en general, a los cambios producidos en la topología de un sistema compuesto (esto es, a cualquier alteración en el número y orden de los nodos y los enlaces que lo constituyen), que es el grado de dinamismo que actualmente se está alcanzando en

### 4.3. Visión arquitectónica de los DHD

---

los Lenguajes de Descripción de Arquitecturas. Según Jeff Kramer, el nombre de Arquitectura Dinámica debería reservarse para referirse a los sistemas en los que incluso el tipo de componentes implicados puede cambiar. Este es, precisamente, el nivel de descripción al que se pretende llegar, por lo que a lo largo de este trabajo esa expresión será utilizada a menudo, sin mayores consecuencias.

....

#### 4.3.8. Tipos de Reconfiguración

Hay varios tipos distintos de reconfiguraciones posibles, que manifiestan problemáticas muy diferentes. Con el fin de determinarlas con mayor exactitud, se han introducido una serie de clasificaciones. En este trabajo se mencionarán dos de ellas, a saber:

- Según el tiempo. Uno de los aspectos críticos en un cambio es el momento en que se produzca. La dificultad para realizarlo de manera adecuada crece según se avanza en el proceso de desarrollo.
  - Antes de la Compilación. Los cambios en la estructura prevista del sistema se producen antes de su construcción, por lo que sólo hay que asegurar su consistencia.
  - Antes de la Ejecución. El sistema ya ha sido elaborado, pero aún no está funcionando, por lo que introducir un cambio sólo requiere, de nuevo, asegurar la consistencia interna del sistema. Puede equipararse sin problemas con el caso anterior.
  - Durante la Ejecución. Esta última es la auténtica Reconfiguración Dinámica: los cambios se producen con el sistema en marcha: los componentes tienen un estado, y sus enlaces soportan cierto número de transacciones. La problemática generada por este tipo de situaciones ha sido el campo de investigación de la comunidad de configuración en los últimos diez años.
- Según el origen. Es decir, según cómo se inicie el proceso; en definitiva, quién toma la iniciativa. Existen dos posibilidades, que son las siguientes:
  - Reconfiguración Programada. Reciben este nombre [EW92, MG99] las operaciones que son iniciadas por el propio sistema, mediante una serie de reglas proporcionadas como parte de su especificación. También se ha mencionado como reconfiguración restringida en tiempo de ejecución [Ore96], y es el nivel de descripción que se desea alcanzar en la Arquitectura de Software.
  - Reconfiguración Ad-Hoc. Además de este nombre [EW92], ha recibido los de reconfiguración evolutiva [MG99] o en tiempo de ejecución

[Ore96]1. Hace referencia a los cambios que son introducidos manualmente por el usuario, y que por tanto requieren en concurso de algún tipo de herramienta, que haga corresponderse a la arquitectura con un sistema real. Obviamente, no pueden ser previstos, aunque en general sí pueden ser controlados

La primera clasificación es obvia, y ha sido planteada entre otros autores por Wermelinger [Wer99]. La segunda se debe inicialmente a los trabajos de Markus Endler en Gerel [EW92], un lenguaje ligado a arwin, pero ha sido redescubierta en múltiples ocasiones por otros autores; esto explica la gran variedad de terminologías utilizadas.

#### 4.3.9. Operaciones Básicas de Reconfiguración

Desde casi el principio, se ha determinado que toda configuración ha de poder ser descrita en función de cuatro operaciones básicas, cuyo establecimiento procede de Conic (§3.6.1). En realidad, no es muy difícil deducirlas de la simple observación de un grafo, considerando en este caso al grafo como la descripción más simple posible de la topología de cualquier sistema, en la que los componentes se expresan como nodos y sus enlaces como aristas.

Las cuatro operaciones mencionadas son las siguientes:

- Crear (create). Se refiere a la creación de nuevos componentes; para ser exacto, la instanciación de un nuevo ejemplar de un tipo de componente conocido
- Destruir (destroy). La inversa de la anterior: permite destruir un componente ya existente; para ser más exacto, eliminarlo del sistema.
- Enlazar (link). Crea un enlace simple entre dos elementos.
- Desligar (unlink). La inversa de la anterior: destruye un enlace preexistente.

No se quiere decir con esto que todo sistema de reconfiguración haya de tener estas cuatro operaciones indicadas de manera explícita; al contrario, muchos las expresan mediante estructuras declarativas, y ni siquiera tienen un equivalente claro para algunas de ellas; lo único que se quiere decir es que toda reconfiguración, por compleja que sea, puede realizarse mediante la aplicación sucesiva de estas cuatro operaciones. Del mismo modo, si un sistema no es capaz de expresar una transformación que, sin embargo, puede ser descrita en función de estas cuatro operaciones, entonces el sistema no es completo, en lo que a este tema respecta.

En realidad, estas operaciones sólo proporcionan lo que aquí se ha denominado dinamismo estructural, ya que únicamente permiten alterar la topología del siste-



#### 4.4. Descripción arquitectónica de los DHD

---

ma. Sin embargo, constituyen lo que podría considerarse el límite mínimo exigible a cualquier sistema que pretenda expresar arquitecturas dinámicas.

En los DHD el dinamismo está basado en la operación denominada Enlazar (link). A lo largo de la revisión quedará claro, no obstante, que hay dos modelos conceptuales que predominan claramente sobre cualquier otro; esto no es solamente cierto en los Adls dinámicos, sino incluso en los convencionales.

#### 4.4. Descripción arquitectónica de los DHD

En esta sección se describe algunos de los aspectos de ArqDHD que tienen que ver con sus atribuciones dinámicas y donde interviene el ContratoDHD. Para ese propósito se tomarán los dos modelos conceptuales creados que mejor se adaptan a ArqDHD. Se trata de los lenguajes algebraicos, fundamentados en álgebras de procesos —si bien en este grupo pueden incluirse también, sin problemas, los enfoques basados en eventos—, y los lenguajes gráficos, esto es, aquellos que se basan en conceptos de teoría de grafos, o incluso teoría de categorías.

##### 4.4.1. Darwin

Existen, en líneas generales, tres versiones del Adl conocido como arwin. La primera, denominada simplemente Darwin, nació como una evolución del ya mencionado Conic, durante el desarrollo del proyecto ESPRIT Rex. Fue elaborada en la primera mitad de los años 90 por Naranker Dulay [Dul90], un miembro del grupo de Kramer, Sloman y Magee.

Aún más importante es la segunda, conocida como arwin 2. En el plano sintáctico es tan sólo un refinamiento de la anterior, pero supuso una reelaboración completa en el plano semántico. En esta etapa el lenguaje fue formalizado en cálculo- [MDEK95], y analizado en contextos concurrentes [Che94]; se elaboraron varias herramientas visuales [Ng92] e interactivas [CDF+95, Fos97], así como un sistema distribuido completo, denominado Regis [MKS94, Cra97]. Incluso se desarrolló una implementación compatible con CORBA [CD97], por entonces más eficiente que algunas comerciales [Cra97]. Esta será, por defecto, la versión a la que se hará referencia siempre que se hable del Adl, ya que es la que ha sido utilizada con mayor frecuencia en trabajos relacionados con la Arquitectura de Software [KM96b, MK96]. Su autor principal fue de nuevo Naranker Dulay, con notorias aportaciones de John Stephen Crane.

La tercera versión, que podría denominarse arwin 3, fue elaborada dentro del proyecto Tracta, y es debida mayoritariamente a Dimitra Giannakopoulou [Gia99]. Se realiza una simplificación notable de la parte estructural, para combinarla después con un modelo de comportamiento completamente nuevo, basado en Sistemas de Transiciones Etiquetadas. En esta versión se fundamentan algunos trabajos

posteriores [MKG99, Pry00], entre los que cabe destacar el magnífico estudio de Moazami-Goudarzi sobre la consistencia frente al cambio [MG99], que continúa la línea iniciada por Jeff Kramer una década antes [KM90].

Por otra parte, existe también una variante de arwin llamada Koala [vOvdLKM00], que es empleada exclusivamente como herramienta de desarrollo en el interior de la compañía Philips.

Aunque tiene sus propias peculiaridades, ya que ha sido especialmente adaptado a la descripción de líneas y familias de productos [JRvdL00], en líneas generales es prácticamente el mismo lenguaje, por lo que no se abundará más en este punto.

arwin ha sido definido como un lenguaje Minimal Orientado a Objetos, Concurrente, de Configuración, de Coordinación, de Composición y, por supuesto, de Descripción de Arquitecturas.

Es uno de los vértices sobre los que se define el propio campo, y representa a una visión muy concreta de sus conceptos que resulta, por ejemplo, completamente opuesta a la de Wright (§3.7.1). De hecho, estos dos lenguajes han sido durante años los ejemplos por excelencia de la visión de la arquitectura de software orientada al objeto –arwin– frente a la arquitectura de software orientada al proceso –Wright– [BF97].

```
component contratoDHD (int n) {  
  provide input;  
  require output;  
  array F[n]:ServForo;  
  forall k:0..n-1 {  
  
    inst F[k];  
    bind F[k].output -- output;  
    when k < n-1 bind  
      F[k].next -- F[k+1].prev;  
    }  
    bind  
    input -- F[0].prev;  
    F[n-1].next -- output;  
  }  
}
```

#### 4.4. Descripción arquitectónica de los DHD

---

##### Una Tubería Extensible en arwin

arwin admite dos notaciones, una textual y otra gráfica. La segunda es más intuitiva, mientras que la primera es, obviamente, mucho más potente. Aquí se considerará exclusivamente esta última. Sintácticamente, todo en arwin es un componente. De hecho, Jeff Kramer es uno de los principales detractores del concepto de conector. Según su visión, tener dos tipos de elementos altera la estructura del sistema, y fuerza a un estilo de descripción en ocasiones poco natural. Las especificaciones de protocolos se indicarán al nivel de un portal, o en un componente específico si resultan demasiado complejas.

Los componentes pueden ser primitivos o compuestos; en el primer caso se corresponden directamente con una implementación. En el segundo, se construyen jerárquicamente a partir de otros componentes. Dado que no hay conectores, tampoco se definen configuraciones; el conjunto total de la arquitectura se ve simplemente como un componente compuesto.

Dentro de un componente jerárquico se realizan dos tipos de enlaces (bindings): entre dos subcomponentes (instancias) del mismo nivel, y entre distintos niveles (enlaces jerárquicos). Se facilita en lo posible la definición de patrones, por lo que se admite la parametrización de los componentes, así como el uso de bucles (forall), condicionales (when), estructuras de tipo array e incluso especificaciones recursivas.

Todo componente tiene unos puntos de interacción definidos, en todo equivalentes al concepto de puerto. De hecho, la versión original de arwin los designaba precisamente con este término. Ahora se denominan portales, y en la vista de comportamiento (ver abajo) su descripción es arbitraria. En cambio, en la versión tradicional siempre han podido ser de dos clases, en las que se definen como asimétricos de manera intencionada: se trata de los servicios (provide) y los requisitos (require). Todo enlace en arwin va desde un requisito hasta un servicio, aunque una vez establecido es completamente bidireccional. Los requisitos expresan solicitudes, mientras que los servicios proporcionan la respuesta a esas solicitudes; por tanto, se admite una cardinalidad de 1 : N. Los portales tienen además asociado un tipo, que indica el esquema de interacción que son capaces de realizar. Los tres básicos son el puerto (port), el flujo (stream) y el evento, aunque es perfectamente posible añadir otros nuevos.

El dinamismo de arwin se manifiesta con claridad en los dos usos de la palabra clave dyn, que define declarativamente dos tipos de reconfiguración: la instanciación perezosa y la instanciación dinámica [MK96, MDEK95]. En el primer caso, un componente es declarado normalmente con su nombre; pero no se crea realmente hasta el momento en que es utilizado.

En el segundo caso, ni siquiera se da nombre al componente, al que sólo se alude mediante su tipo. Se crea dinámicamente en el momento en que otro componente intenta activar uno de sus enlaces; y él mismo genera un nuevo enlace, desde el que tomará toda iniciativa de comunicación en adelante –los demás no

pueden invocarlo, ya que no conocen un nombre con el que hacerlo—. En definitiva, el dinamismo de arwin consiste en un tipo de reconfiguración restringida, que expresa unos patrones fijos, pero muy potentes, en los que se admite la creación de componentes y enlaces. Por otra parte, la evolución dinámica de los enlaces produce de manera habitual alguna reconfiguración trivial (al respecto, revisar el enfoque basado en cálculo-, que se describe más abajo). Con todo, su versatilidad sigue siendo inferior a la de Conic. Aparte de esto, en [KM96b] se propone la definición de arquitecturas auto-organizadas, en las que una serie de restricciones globales (replicadas localmente en cada componente) controlan la evolución del sistema, sin que tenga que haber un elemento específico que las aplique. Este enfoque, muy citado en la Bibliografía, no ha sido sin embargo explorado desde entonces.

Uno de los aspectos más interesantes de arwin ha sido la especificación formal de la semántica de sus enlaces en cálculo-. Esto abarca tanto a los enlaces “estáticos” como a las reconfiguraciones dinámicas citadas [MDEK95, MK96, Cra97]. Como muestra, se indica en la Figura 3.5 la semántica de un requisito, un servicio y un enlace simple, y se indica cómo interaccionan. Tras una evolución inicial (el establecimiento del protocolo), puede comprobarse que los distintos procesos quedan ligados (cada uno de ellos conoce ya a los canales del otro), mientras el servicio (Prov) mantiene la capacidad de seguir atendiendo peticiones.

Prov(p, s) def = !(p(x).xs) Req(r, o) def = r(y).yo Bind(r, p) def = rp Semántica de un servicio Semántica de un requisito Semántica de un enlace

( Req(r, o) — Bind(r, p) — Prov(p, s) ) . . . r . . . p os — Prov(p, s)

Figura 3.5: Establecimiento de un Enlace arwin en cálculo-

Como ya se ha dicho, la última versión de arwin parece haber derivado más hacia la parte de especificación. A la perspectiva de este nuevo enfoque se le denomina vista de comportamiento, y a la tradicional, vista de construcción. La descripción de los protocolos ha abandonado parcialmente el cálculo- en favor de los sistemas de transiciones etiquetadas (LTS), que facilitan la construcción de herramientas de verificación, como el Labelled Transition Systems Analyzer (LTSA) [MKG97, Gia99]. Su mayor problema, la explosión de estados, se controla mediante una técnica desarrollada por Jeff Kramer y Shing-Chi Cheung, el análisis de alcanzabilidad composicional (CRA). Por su parte, Kaveh Moazami-Goudarzi ha utilizado esta técnica para comprobar los estados del sistema antes y después de una reconfiguración [GK96, MG99]. Aunque el sistema de LTS tiene la ventaja de su simplicidad, parece no obstante demasiado limitado para llegar a plantearse en sistemas de cierto tamaño. En resumen, arwin es uno de los Adls más potentes y flexibles. Se puede utilizar en varios niveles de descripción, y a pesar de no ser formal, tiene una semántica formalizada. Sus mayores defectos son precisamente su relación ambigua con los aspectos formales, su habitual dependencia de las distintas versiones de Regis, y el hecho de tener un dinamismo limitado, aunque considerable, en su versión más declarativa.

### 4.4.2. Wright Dinámico

El lenguaje Wright [All97] es uno de los Adls más conocidos, y posiblemente el que tiene la sintaxis más formal de todos los existentes. También es el más estático de todos ellos [CFBS99]. Plantear la presencia de este lenguaje en un Capítulo sobre Arquitectura Dinámica parece casi una contradicción, pero en efecto existe una propuesta para una versión de Wright Dinámico [ADG97, ADG98]. Una descripción Wright convencional consiste en una configuración compuesta por una serie de componentes, cuyo interfaz está segmentado en puertos. Los distintos puertos se relacionan entre sí, internamente, a través de la computación definida por el componente. Por su parte, la comunicación entre componentes requiere la existencia de conectores, cuyo interfaz se segmenta en roles, relacionados entre sí mediante un pegamento (glue). La combinación de puertos y computación especifica las posibilidades de comunicación de un componente; la de roles y pegamento, el protocolo proporcionado por un conector. La unión del puerto y el rol adecuado recibe el nombre de adjunción(attachment) y es lo que hace posible la comunicación.

Todos estos detalles son descritos con el lenguaje Csp, un álgebra de procesos [Hoa85]. Además, Wright es uno de los pocos lenguajes que admite también la especificación de estilos, mediante el uso de una variante de la Lógica de Predicados.

La descripción de Wright Dinámico es en principio completamente idéntica, y tan sólo añade un detalle, que es el que sigue: en todo sistema dinámico se asume la existencia de un componente especial denominado *Configurador*, que se encargará de realizar las reconfiguraciones requeridas por el sistema.

Para poder hacerlo, el lenguaje es ampliado con las cuatro operaciones básicas de la reconfiguración, que aquí reciben los nombres de *new*, *del*, *attach* y *detach*, y cuya correspondencia con las originales resulta evidente. Ha de tenerse en cuenta que estas operaciones sólo pueden usarse desde el Configurador, ya que éste elemento será el que contenga la especificación completa de los aspectos dinámicos del sistema.

La Figura ?? proporciona parte de la definición de un Configurador para un ejemplo donde hay un contrato interviniendo entre un usuario y el servicio brindado por el método *getTheme* del Foro. En ?? se presenta una metodología para la aplicación que permite calcular y describir el lugar donde se debe poner el ContratoDHD, mediante la identificación de la componentes y sus métodos

La parte que no se muestra proporciona la definición del proceso *Coordination*, en las que se encuentran todas las actividades de coordinación del contrato.

```
Configurador DynamicContractDHD
  Style CondicionalesDEV
    new.T:Foro
```

```
new.Cdhd : ContratoDHD
new.C : Athour
attach.T.p.getTheme.Cdhd.c Coordination
where . . .
```

Para este caso, usando wright dinámico, el evento *getTheme*, que se comunica desde el puerto *p* del componente *T* (un Foro), hasta el rol *r* del conector *Cdhd* (un ContratoDHD) es manejado por el sistema como el evento *T.getTheme.p1.Cdhd1.r1*; y un componente es identificado por su nombre, seguido de la especificación de la conexión de todos sus puertos, de modo que se pueda saber en qué configuración concreta se encuentra. Es decir, el componente *T* será, en un momento dado, el *T..p1.Cdhd1.r1 . . . pn.Cdhdn.rn*, donde los distintos *pi* son sus puertos, y los *ri* los roles de los conectores Ni a los que éstos están conectados.

Por tanto, las operaciones *new*, *del*, *attach* y *detach* tan sólo cambian el etiquetado; un componente no se crea, sino que se le da la etiqueta adecuada; ni tampoco se destruye, sino que simplemente se le da un prefijo diferente.

## 4.5. Conclusiones

completar ...

#### 4.5. Conclusiones

---

---

## Contratos sensibles al contexto

---

### 5.1. Introducción

### 5.2. Hacia la definición de ContratoDHD

La componente contrato es la información que se tiene de una componente. El contrato puede ser configurado por medio de diferentes mecanismos, desde el lenguaje cotidiano hasta un lenguaje de especificación formal y un lenguaje basado en XML <sup>a</sup> para los casos que sean necesarias especificaciones que puedan ser procesadas por máquinas.

#### 5.2.1. Características que debe cumplir el ContratoDHD

El tipo de tecnología y forma de implementación de los contratos es transparente para los objetos que consumen los servicios en donde se encuentran involu-

---

<sup>a</sup>XML, sigla en inglés de eXtensible Markup Language («lenguaje de marcas extensible»), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML que permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.



## 5.2. Hacia la definición de ContratoDHD

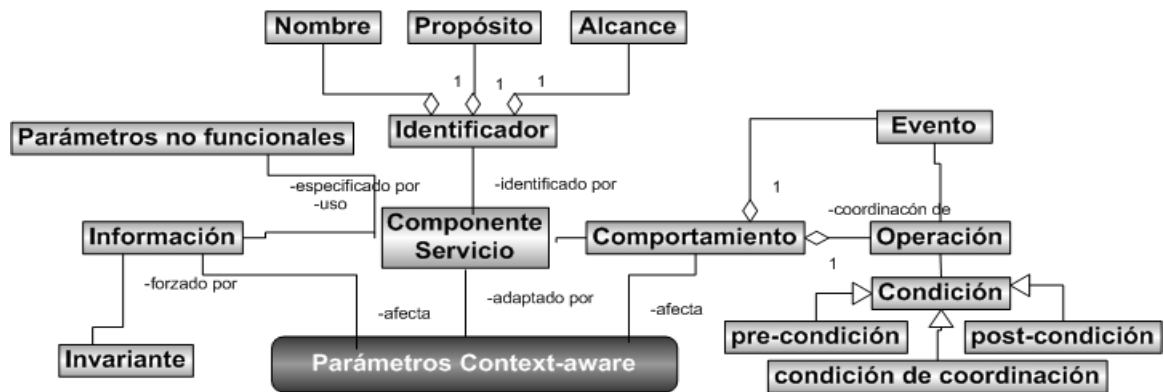


Figura 5.1: Caracterización del ContratoDHD y sus Relaciones

crados.

La configuración de un elemento contrato que forma parte de las componentes de un servicio, representa la información necesaria del mismo para ser utilizado por el invocador, sin necesidad que el objeto invocador tenga detalles de la ejecución.

El contrato representa una enriquecida y efectiva interface de construcción que contiene toda la información sobre las componentes de los servicios y deberá tener información sobre algún tipo de información de contexto para su utilización.

El concepto de interface en este caso, tiene mayor significación que un simple acceso a un contrato de reglas entre el objeto proveedor del servicio y el objeto consumidor. En la figura ?? podremos observar, los elementos conceptuales básicos de esta componente a través de una serie de elementos en relación con el contrato; este metamodelo tiene las características conceptuales y operativas que se fundamentan en Obra Abierta.

Para una mejor comprensión de las componentes del modelo explicitaremos a continuación su caracterización y funciones particulares:

- **Identificador.**

Una componente servicio es identificada para un determinado contexto por un único nombre en el espacio de nombre.

- **Comportamiento.**

De acuerdo con los roles asignados en un determinado contexto, una componente servicio expone comportamientos correspondientes a provisión y pedido de operaciones, y/o publicaciones y recepción desde/hacia cada contexto. Las operaciones pueden ser definidas en dos tipos – operaciones que ejecutan cálculos o transformaciones (tipo “update”) y operaciones que proveen algún tipo de información sobre consultas (tipo “query”). Estas, se encuentran enteramente especificadas en base a un contrato, con el uso de pre-condición, post-condición y condicionales para lograr la coordinación

entre contratos. En las condiciones de coordinación se especifican cómo requerir y proveer operaciones, así como también los eventos publicados y recibidos son coordinados en los momentos adecuados. Para lograr una comunicación precisa con una componente servicio, no sólo se tiene en cuenta qué operación fue provista o requerida y cómo el ejecutor ha lanzado el evento apropiado, sino también, cómo todas esas actividades están mutuamente relacionadas para ser aprovechadas por el objeto consumidor. Un evento del contexto que lanza una operación dada, puede ser parte de un conjunto de pre-condición, mientras que un evento emitido a través de una exitosa operación puede ser parte de una pos-condición.

Las operaciones provistas y requeridas por la componente de servicio deben estar asociadas, a fin de determinar las operaciones que deben ser completadas antes de la activación de un servicio (qué es posible de ejecutarse en paralelo o ser sincronizado por otro camino). Por ejemplo, en el caso de una componente de servicio *ManegadorOrden*, la operación *HacerOrden* no puede ser invocada hasta que el servicio que la consume no esté correctamente autorizado por el componente de servicio *AdministradorRegistros*, o la operación *deleteOrder* no puede ser invocada si la operación *HacerOrden* con el mismo parámetro *OrdenID* no fue previamente completada.

- *Tipos de Información.*

Una componente de servicio debe manejar, usar, crear o tener cierta información de recursos con el propósito de proveer servicios adecuadamente. Este elemento del contrato define el tipo de información relevante para las componentes asociadas al contrato, así como también restricciones y reglas sobre instancias de esos tipos. Esto representa un modelo de información lógica de una componente de servicio. Formalmente, esta información de tipos puede ser considerada como definiciones de tipos de los parámetros de las operaciones o tipos relacionados a ellos.

- *Configuración de Parámetros Context-Aware*

Una componente servicio depende del contexto de su actual entorno. La misma, para utilizarse en diferentes contextos logrando la adaptación a eventuales cambios debe tener definido un conjunto de parámetros de configuración. Ejemplos de estos parámetros pueden ser: *Contexto-del-Usuario (CU)* - en un sentido similar a lo definido en el capítulo anterior, *locación en tiempo y espacio* de los servicios consumidos y suministrados. Estos parámetros, pueden ser enviados dentro de las invocaciones de las operaciones de los servicios o por medio de otros caminos, mediante componentes de servicios que pueden adaptar su comportamiento ante el cambio de contexto en una determinada situación.

La configuración de parámetros está directamente asociada a las relaciones de las operaciones de los servicios para lograr una mejor adaptación a la medida de las circunstancias brindada por la información relevada del contexto. El concepto de la configuración de los parámetros context-aware, es un paso muy importante hacia la concepción de servicios automatizados y auto adaptables (tomando el sentido paradigmático de los teóricos de la Inteligencia Artificial).

- *Parámetros no funcionales.*

### 5.3. Hacia la definición de contratos para el DHD

---

Una componente servicio puede definir un conjunto de los llamados parámetros no funcionales que caracterizan a la “calidad” de sus prestaciones dentro de un determinado contexto. Estos parámetros, son elementos para los consumidores de los servicios que permiten optar por el uso de un determinado servicio, o buscar otro con el mismo o similar contrato. Como ejemplo de parámetros no funcionales podemos mencionar: Performance, Fiabilidad, Tolerancia a Fallos, Costos, Prioridad y Seguridad.

### 5.3. Hacia la definición de contratos para el DHD

Una vez más se abordará una definición teniendo en cuenta cuestiones tecnológica inherentes a su implementación. La implementación de los contratos para el DHD debe cumplir el diseño de la figura 5.1 a través de la metodología del agregado del framework JCA<sup>b</sup> (véase forma de agregado en la sección ??)

### 5.4. ContratosDHD: Contratos sensibles al contexto

Para esta tesis se han explorado diferentes implementaciones de los contratos de Meyer, preferentemente aquellas que se puedan ajustar mejor en nuestro desarrollos y diseño. De esta manera se tuvieron en cuenta avances donde el uso de contrato para la representación de aserciones permitan una buena integración para el DHD. Desafortunadamente, existen pocos lenguajes que soportan técnicas de aserciones (ej. Eiffel<sup>c</sup>). Las forma de implementación y la tecnología (e.j., lenguajes de programación, etc.) que se utiliza tienen una alta incidencia en el diseño y arquitectura. En este sentido, se analizaron tres diferentes tipos de estrategias de implementación.

- *Built in*: significa que el soporte de los contratos está incluido en los el lenguaje de programación. Donde se tienen constructores de lenguajes para la formulación de aserciones de diferentes formas. La corrección de la sintaxis de la aserción es directamente chequeada por el compilador. Además, el entorno de ejecución posibilita que se controle dicho chequeo en tiempo de ejecución. Las mayores ventajas de estas implementaciones tienen que ver con la homogenia integración de las aserciones dentro del lenguaje de programación, i.e., los mensajes de error son consistentes, las herramientas

---

<sup>b</sup>JCAF is designed to support Context-Aware Computing and has come out of our work with the design of context-aware applications in a hospital environment.

<sup>c</sup>Eiffel fue ideado en 1985 por Bertrand Meyer. Es un lenguaje de programación orientado a objetos centrado en la construcción de software robusto. Su sintaxis es parecida a la del lenguaje de programación Pascal. Una característica que lo distingue del resto de los lenguajes es que permite el diseño por contrato desde la base, con precondiciones, postcondiciones, invariantes y variantes de bucle, invariantes de clase y aserciones.

de depuración permiten un adecuado manejo e implementación de aserciones (ej., correctitud de número de línea y traza de ejecución).

- *Preprocessing*: Esta es la clase mas popular de soporte para las aserciones en los lenguaje de programación. La idea general es la formulación de aserciones separadas del programa o incluirlas como comentarios. Utiliza un pre-proceso para entretejer aserciones entre el programa o transformar los comentarios en porciones de código que la ejecuten. La principal ventaja de este método es la separación entre el contrato y la lógica de programación.
- *Metaprogramming*: De acuerdo con Temp1 la metaprogramación refiere a "programación para los niveles de la interpretación de los programas, o en otras palabras, para extender las interpretaciones de un lenguaje de programación hacia una eplicación específica" (137). Programas que tienen la posibilidad de razonar sobre si mismo tiene la propiedad llamada reflectiva (en inglés "reflective capacity"). Por ejemplo, el lenguaje de programación Java tiene capacidades reflectivas y es implementada a través de una API. La mayor ventaja de la metaprogramación es que no es necesario el uso de preprocesamiento. No obstante, un entorno de ejecución especializado debe ser usado para el chequeo de las aserciones.

### Sistema que proveen soportes para el desarrollo de software basado en contratos

Se analiza brevemente diferentes sistemas que proveen soporte para el desarrollo de software basado en contratos a través del lenguaje de programación Java. Estas propuestas fueron estudiadas en base a los prototipos disponibles para evaluar el impacto en nuestra particular implementación, i.e., no solamente una evaluación teórica. No fueron consideradas propuestas que usan técnicas de álgebraicas o lógica de alto orden. De la misma manera sistemas como JML (138) no fueron considerados.

A continuación se detalla para cada propuesta de sistema el nombre, el tipo de resolución implementada (lenguajes de programación, extensiones de lenguajes, preprocesamiento, metaprogramción), consideraciones generales, características especiales y referencias.

- Biscotti:
  - Tipo de soporte: Extensión del lenguaje Jaca; tiene incorporado un soporte para compilación.
  - Información: Biscotti se concentra en la implementación de especificaciones de comportamiento a través de interfases Java introduciendo palabras claves adicionales.
  - Características especiales: No es necesario hacer cambios para la máquina virtual java, esto es posible debido a que Biscotti usa reflexión y

## 5.4. ContratosDHD: Contratos sensibles al contexto

---

facilita la creación de precondiciones, postcondiciones e invariantes visibles en tiempo de ejecución.

- References: (141)

### ■ Java Assertion Facility (JAF):

- Tipo de soporte: Implementado desde la versión 1.4 de Java.
- Información: Desde la versión 1.4 de Java se incorporó palabra claves para aserciones para indicarles al compilador donde se encuentran las aserciones y su interpretación a través de la máquina virtual. Java solo soporta mecanismos de aserciones simples que permiten instrumentar condiciones a través de métodos.
- Características especiales: El chequeo de las aserciones pueden ser fácilmente habilitadas y deshabilitada y los rastros de las aserciones pueden ser completamente eliminados de los archivos de las clases.
- Referencias:(148; 146; 147).

### ■ ContractJava:

- Tipo de soporte: ContractJava es el típico preprocesamiento en el que se genera código Java.
- Información: El preprocesamiento soporta los usuales tipos de precondiciones, postcondiciones y aserciones.
- Características especiales: Soporta comportamiento de subtipo (Behavioral subtyping)(139)
- Referencias: (140)

### ■ iContract:

- Tipo de soporte: Preprocesamiento en el que se genera código Java para ser compilado por el compilador estándar Java.
- Información: Las aserciones son escritas como documentación de métodos y clases
- Características especiales: iContract soporta operaciones sobre colecciones y provee herramienta adicionales para la documentación de aserciones en Java Doc (iDoclet) y para facilitar el rediseño de clases Java (iDarvin)
- Referencias: (142; 143)

### ■ Jass:

- Tipo de soporte: Preprocesador para la generación de código Java antes de la compilación.

- Información: Los significados de las aserciones son especificados por medio de comentarios para métodos y clases. Un programa Jass es un programa Java bien formado y puede ser transformado directamente para el compilador estándar Java.
- Características especiales: Jass soporta cuantificadores universales y existenciales para conjuntos finitos.
- Referencias: (151)

### ■ Jcontract:

- Tipo de soporte: Preprocesador para la generación de código java para el compilador estándar Java.
- Información: Las aserciones se especifican como comentarios para métodos y clases.
- Características especiales: Jcontract soporta cuantificadores existenciales y universales para tipos de los colecciones. Además, Jcontract está estrechamente integrado con la herramienta Jtest(145). Jtest la información de especificación contenida en el contrato, luego crea una clase de test ejecutable que permite testear si se cumple la funcionalidad de la clase especificada. Incluye herramienta de generación de javadoc para la generación de documentación API donde contiene las especificaciones de las aserciones.
- Referencias:(144)

### ■ Handshake:

- Tipo de soporte: Provee aserciones a través de reflexión (Java reflection).
- Información: Los contratos son separados en archivos diferentes. Las clases son combinadas en tiempo de ejecución combinando dinámicamente los archivos de los contratos y las clases Java.
- Características especiales: Debido a la propiedades dinámicas los contratos son incorporados a las clases a nivel de código de ejecución.
- Referencias:(150)

### ■ jContractor:

- Tipo de soporte: jContractor se basa puramente en librerías base utilizando metaniveles de información deducidas de los archivos de las clases Java y usa la ejecución dinámica de las clases Java a través de la interpretación de cambios reflectivos en tiempo de ejecución.
- Información: La codificación del contrato (precondiciones, postcondiciones e invariantes) son incorporados a las clases por medios del especificados con determinadas convenciones de nombres.

## 5.4. ContratosDHD: Contratos sensibles al contexto

---

- Características especiales: Dentro de las postcondiciones de un método es posible controlar los resultados esperados con otros resultados específicos.
- Referencias: (144)

**Definición 6** (ContratoDHD). *El ContratoDHD una aplicación de los contratos de Meyer (70) según la filosofía "Design by Contract"<sup>d</sup> con el agregados de tres componentes esenciales para los contratos (CEC) conformadas por subsistemas y sus forma de conexión:*

- *CEC #1: Componentes del contratos para la implementación de las propiedades de sensibilidad al contexto. Los elementos que integran este subsistema son:*
  - *Arquitectura para la adaptación del contextos*
  - *Funcionalidades*
- *CEC #2: Componentes de conexión e implementación de un patrón de diseño que permite la coordinación de contratos.*
  - *Reglas de coordinación*
    - *Condicionales especiales*
    - *Condicionales comunes*
  - *Componentes de conexión*

### 5.4.1. Componentes esenciales de los contratos sensibles al contexto

En este apartado, expondremos los diferentes tipos de componentes para capturar contexto que integran un modelo context-aware. Particularmente, mencionaremos uno de los más tradicionales expuesto en la tesis doctoral de Dey, A. (2000). A continuación se muestra una clasificación de arquitecturas en las que se resaltan las principales características.

---

<sup>d</sup>The term was coined by Bertrand Meyer in connection with his design of the Eiffel programming language and first described in various articles starting in 1986 and the two successive editions (1988, 1997) of his book *Object-Oriented Software Construction*. Eiffel Software applied for trademark registration for Design by Contract in December 2003, and it was granted in December 2004.

- *Servidor de contexto*: ✓ Permite el acceso de múltiples clientes a los datos de forma remota. ✓ Libera a los clientes de recursos que demandan operaciones intensivas. ✓ Debe considerar el uso de protocolos apropiados, rendimiento de la red, calidad de los parámetros de los servicios.
- *Widgets*: ✓ Encapsulamiento. ✓ Intercambiable. Controlado a través de un manejador de widget. ✓ En el caso que las componentes estén acopladas incrementan la eficiencia al costo de perder robustez ante fallas de componentes.
- *Networked services*: ✓ Similar a la arquitectura servidor de contexto. (modelo orientado a objeto). ✓ tiene la eficiencia de la arquitectura *widget* debido a la complejidad de las componentes bases de red, pero provee robustez.
- *Blackboard model*: ✓ Procesos post-mensajes para compartir medios, pizarrón (modelo basado en datos). Simplicidad en el agregado de nuevos recursos de contexto. ✓ Fácil configuración. ✓ Un servicio centralizado. ✓ Carece de eficiencia en la comunicación (son necesarios dos puntos de conexión por comunicación)

Llegados a este punto, nos introduciremos en una definición de la componente principal del modelo: el Widget.

Un *widget* es un componente de software que brinda una interfaz pública para algún tipo determinado de sensores (hardware) (Dey, A. K. y G. D. Abowd, 2000). Los *widgets* ocultan detalles de censado de bajo nivel y al mismo tiempo son componentes con alto grado de reusabilidad, facilitando el desarrollo de aplicaciones. El encapsulamiento en los *widgets* permite intercambiarlos entre aquellos que proveen el mismo tipo de datos (ejemplo: intercambiar un *widget* de radiofrecuencia por un *widget* de una cámara filmadora donde ambos recolectan datos de locación de individuos). Usualmente los *widgets* son controlados por alguna clase de administrador de *widget*. En el caso que las componentes sean acopladas incrementan la eficiencia, al costo de perder robustez ante fallas de componentes.

Se identifican cinco categorías de componentes que implementan distintas funciones:

- *Context Widgets*: se encarga de la adquisición de información de contexto;
- *Interpreters*: cumplen la función de transformar y aumentar el grado de abstracción de la información de contexto, deben combinar varias piezas de contexto para producir información procesada de alto nivel;
- *Aggregator*: es un componente que junta información de contexto de una entidad para facilitar su acceso a las aplicaciones;
- *Services*: brindan servicios en un determinado ambiente adaptando su funcionalidad al tipo de información de contexto adquirida;



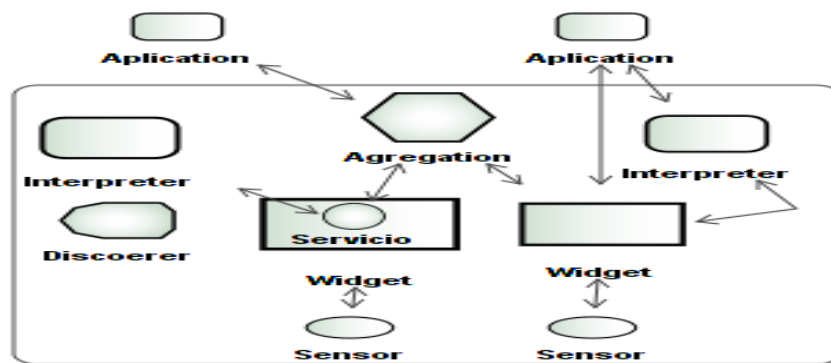


Figura 5.2: Elementos para la toma de contexto en los ContratosDHD

- *Discoveres*: permite que las aplicaciones (y otras entidades) optimicen su desempeño pudiendo determinar la característica del entorno (tipo de restricciones y dominio de aplicación). Entre estos componentes se pueden establecer un número limitado de relaciones.

Widgets es consultada, o bien, notificada ante eventuales cambios en los clientes. Los clientes pueden ser *aplicaciones*, *aggregators* u otros *widgets*. A su vez, *aggregators* actúa como un puente entre *widgets* y aplicaciones. Un *interpreters* puede ser solicitado en un determinado estado por un *widget*, *aggregator* o *aplicaciones*. Los *services* son lanzados por las aplicaciones (también otros componentes pueden hacer uso de los servicios). *Discoveres* se comunica con todos los componentes, se adquiere desde los *widget*, *interpreters* y *aggregators*, y provee información a las aplicaciones por medio de notificaciones y consultas.

La figura 5.2 expone una posible configuración con dos dispositivos de censado, dos *widgets*, un *aggregator*, dos *interpreters*, un *servicio*, un *discoverer* y dos *aplicaciones*.

En el instante en que algún componente contexto se encuentre disponible, registra sus capacidades en un *discoverer*. Esto permite que los *aggregators* encuentren *widgets* e *interpreters* y, a las aplicaciones, encontrar *aggregators*, *widget* e *interpreters*. Un sensor provee datos para un *context widget*, el cual se encarga de almacenar el contexto. Puede llamar a un *interpreter* para obtener un mayor nivel de abstracción de los datos y luego pone el contexto a disposición (para que pueda ser accedido) por otros componentes y aplicaciones.

Un *aggregator* recolecta información de contexto de las entidades, representadas por los *widgets*. Finalmente, las aplicaciones pueden consultar o suscribirse a los *agregators* (o directamente con los *widgets*, si se quiere) y llamar a *interpreters* (si el deseado nivel de abstracción no se encuentra disponible desde los *widgets* y *aggregators*).

Estos componentes se ejecutan independientemente de las aplicaciones, ase-

gurando una continua adquisición de información de contexto y el uso de múltiples aplicaciones. Además, todos los componentes y aplicaciones se comunican entre ellos automáticamente usando protocolos y lenguajes conocidos.

Esto da lugar a que los programadores que implementan un particular componente o aplicación puedan comunicarse con otro componente sin tener conocimiento de los mecanismos usados para lograr la interacción.

### 5.4.2. Una implementacio del subsistema para CEC#1

La implementacion del subsistema de las componentes esenciales para el agregado de propiedades de sensibilidad al contexto (CEC#1) tienen dos significancia para esta tesis. En primer lugar, es brindar un ejemplo concreto de integracion utilizando parte del recorrido teorico y documental de tipos de sistemas (secciones 5.4) y estrategias (secciones 5.4). En segundo lugar, presentar nuevos conceptos y definiciones que permitan generalizar este tipo de propuestas de comunicacion entre sistemas.

Para este caso se utiliza como subsistema para CEC#1 el framework JCAF (*Java Context-Awareness Framework*). A través de JCAF se implementan la ejecución de servicios (servicios de herramientas colaborativas) por medio de la infraestructura orientada a servicios basada en la idea de la division de la adquisición de contexto, su manejo y distribucion o conexión con los ContratosDHD. Además JCAF provee un modelo de programacion para Java extensible, generico y expresivo orientado al desarrollo e implementación de aplicaciones sensibles al contexto y modelado de contexto. En la publicacion (149) se brindan mas detalles sobre como JCAF difiere de otros soporte middleware para aplicaciones sensibles al contexto.

Para describir mas apropiadamente la participacion que JCAF tiene para nuestra propuesta de implementacion es necesario determinar cuales de los aspectos orientados a servicios seran recogidos. De esta manera se brinda una deficion sobre el contexto en el que se tienen en cuenta los servicios.

**Definición 7** (ServiciosDHD:). *Se denomina servicioDHD a cualquier implementacion sobre las funcionalidades de una herramienta, de un espacio colaborativo, (1) que tenga interfases de comunicacion con la componente parametros Context-Aware (figura ??) y (2) tiene su implementacion contiene elementos que representan entidades participantes e informacion de contexto (66). En otras palabras, la implementacion de un servicioDHD esta compuesta por modulos para la representacion y uso contexto y que los mecanismos de ejecucion esten mediado por los ContratosDHD*

## 5.4. ContratosDHD: Contratos sensibles al contexto

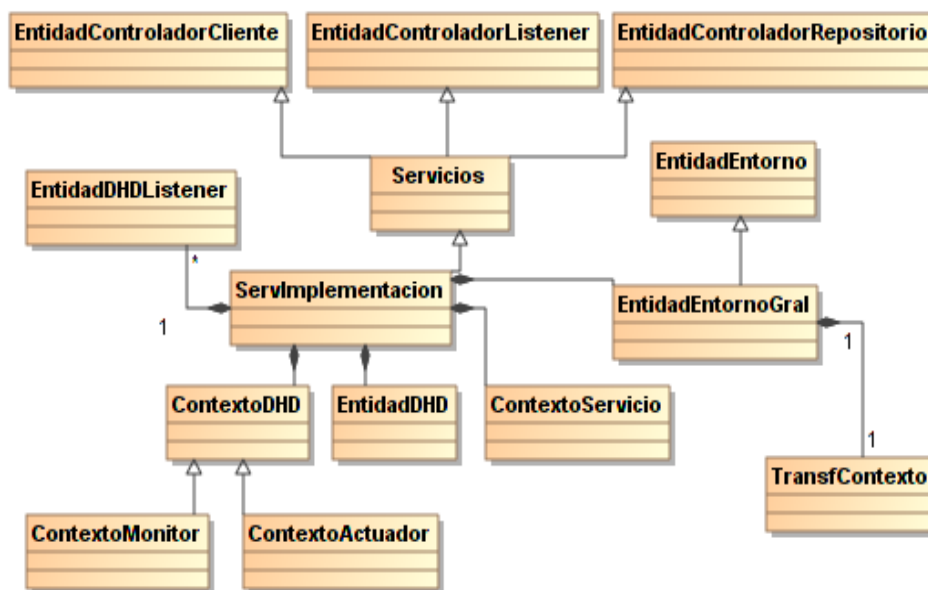


Figura 5.3: Arquitectura de ejecución de serviciosDHD

### Infraestructura para serviciosDHD

A partir de la infraestructura que brinda JCAF para combinar modelos e implementaciones de servicios con información de contexto, se despliega una serie de articulaciones que permita una adaptación para el uso de contratos. De esta manera quedará conformada una nueva infraestructura para los serviciosDHD.

La parte más importante

#### 5.4.3. Coordinación de los contratos sensibles al contexto

Elementos de la componente contrato La componente contrato es la información que se tiene de una componente. El contrato puede ser configurado por medio de diferentes mecanismos, desde el lenguaje cotidiano hasta un lenguaje de especificación formal y un lenguaje basado en XML2 para los casos en que sean necesarias especificaciones que puedan ser procesadas por máquinas.

El tipo de tecnología y forma de implementación de los contratos es transparente para los objetos que consumen los servicios en donde se encuentran involucrados. La configuración de un elemento contrato que forma parte de las componentes de un servicio representa la información necesaria del mismo para ser utilizado por el invocador, sin necesidad de que el objeto invocador tenga detalles de la ejecución.

El contrato representa una enriquecida y efectiva interfaz de construcción que

contiene toda la información sobre las componentes de los servicios y deberá tener información sobre algún tipo de información de contexto para su utilización.

En este caso, el concepto de interfaz tiene mayor significación que un simple acceso a un contrato de reglas entre el objeto proveedor del servicio y el objeto consumidor.

En la figura 2 podemos observar los elementos conceptuales básicos de esta componente a través de una serie de elementos en relación con el contrato. Este metamodelo tiene las características conceptuales y operativas que se fundamentan en “Obra abierta”.

Para una mejor comprensión de las componentes del modelo explicitaremos a continuación su caracterización y funciones particulares.

**Identificador:** una componente servicio es identificada para un determinado contexto por un único nombre en el espacio de identificación.

**Comportamiento:** de acuerdo con los roles asignados en un determinado contexto, una componente servicio expone comportamientos correspondientes a provisión y pedido de operaciones, y/o publicaciones y recepción desde/hacia cada contexto. Las operaciones pueden ser definidas de dos tipos: operaciones que ejecutan cómputo o transformaciones (tipo update) y operaciones que proveen algún tipo de información sobre consultas (tipo query). Éstas se encuentran enteramente especificadas en base a un contrato, con el uso de precondition, poscondition y condicionales para lograr la coordinación entre contratos. En las condiciones de coordinación se especifican cómo requerir y proveer operaciones, así como también los eventos publicados y recibidos son coordinados en los momentos adecuados. Para lograr una comunicación precisa con una componente servicio, no sólo se tiene en cuenta qué operación fue provista o requerida y cómo el ejecutor ha lanzado el evento apropiado, sino también cómo todas esas actividades están mutuamente relacionadas para ser aprovechadas por el objeto consumidor. Un evento del contexto que lanza una operación dada puede ser parte de un conjunto de precondition, mientras que un evento emitido a través de una exitosa operación puede ser parte de una poscondition.

Las operaciones provistas y requeridas por la componente de servicio deben estar asociadas, a fin de determinar las operaciones que deben ser completadas antes de la activación de un servicio (qué es posible ejecutar en paralelo o ser sincronizado por otro camino). Por ejemplo, en el caso de una componente de servicio `ManejadorOrden`, la operación `HacerOrden` no puede ser invocada hasta que el servicio que la consume no esté correctamente autorizado por el componente de servicio `AdministradorRegistros`, o la operación `DeleteOrder` no puede ser invocada si la operación `HacerOrden` con el mismo parámetro `OrdenID` no fue previamente completada. Tipos de información: una componente de servicio debe manejar, usar, crear o tener cierta información de recursos con el propósito de proveer servicios adecuadamente. Este elemento del contrato define el tipo de información relevante para las componentes asociadas al contrato, así como también

restricciones y reglas sobre instancias de esos tipos. Esto representa un modelo de información lógica de una componente de servicio. Formalmente, esta información de tipos puede ser considerada como definiciones de tipos de los parámetros de las operaciones o tipos relacionados a ellos. Configuración de parámetros context-aware: una componente servicio depende del contexto de su actual entorno. La misma, para utilizarse en diferentes contextos logrando la adaptación a eventuales cambios, debe tener definido un conjunto de parámetros de configuración. Ejemplos de estos parámetros pueden ser: Contexto del Usuario (CU), en un sentido similar a lo definido en el capítulo anterior, locación en tiempo y espacio de los servicios consumidos y suministrados. Estos parámetros pueden ser enviados dentro de las invocaciones de las operaciones de los servicios o por medio de otros caminos, mediante componentes de servicios que pueden adaptar su comportamiento ante el cambio de contexto en una determinada situación. La configuración de parámetros está directamente asociada a las relaciones de las operaciones de los servicios para lograr una mejor adaptación a la medida de las circunstancias brindada por la información relevada del contexto. El concepto de la configuración de los parámetros context-aware es un paso muy importante hacia la concepción de servicios automatizados y autoadaptables (tomando el sentido paradigmático de los teóricos de la inteligencia artificial).

**Parámetros no funcionales:** una componente servicio puede definir un conjunto de los llamados parámetros no funcionales que caracterizan la “calidad” de sus prestaciones dentro de un determinado contexto. Estos parámetros son elementos para los consumidores de los servicios que permiten optar por el uso de un determinado servicio, o buscar otro con el mismo o similar contrato. Como ejemplo de parámetros no funcionales podemos mencionar: performance, fiabilidad, tolerancia a fallos, costos, prioridad y seguridad.

### Coordinación

En términos generales, la coordinación de contratos es una conexión establecida entre un grupo de objetos (en nuestras consideraciones los participantes serían un objeto cliente y un determinado servicio), donde reglas, normas y restricciones (RNR) son superpuestas entre los actores participantes, estableciendo con un determinado grado de control las formas de interrelación (o interacción).

El tipo de interacciones establecidas entre las partes es más satisfactoria que las que se pueden lograr con UML o lenguajes similares (orientados a objetos) debido a que éstas contienen un mecanismo de superposición donde se toman como argumento los contextos. Cuando un objeto cliente efectúa una llamada a un objeto suministro, el contrato “intercepta” la llamada y establece una nueva relación teniendo en cuenta el contexto del objeto cliente, el del objeto servidor e información relevante (respecto de la relación) adquirida y representada como contexto del entorno. Como condición necesaria, la implementación de los contratos no debe alterar el diseño y funcionalidad en la implementación de los objetos.

### 5.5. Condicionales especiales para las reglas de los contratos

En las implementaciones de campo de cursos de nivel superior realizadas por el equipo de “Obra abierta”, uno de los recursos primarios que los educadores utilizaban era el registro de actividades de la plataforma. Podemos ahora considerar una información de contexto, a un valor cuantitativo que resignifique una característica del comportamiento de un usuario, tal como lo expusimos en el ??.

Consultar los registros de actividades es una práctica muy usual para obtener información sobre el “grado” de interactividad que los usuarios tienen en el marco del dispositivo hipermedial. En los casos más básicos, existe una estructura de “planilla” (filas y columnas) en las que se detallan los registros con los siguientes campos: fecha, hora, usuario, tareas realizadas. Las tareas pueden estar significadas con valores literales como: “visitas al foro”, “visualización de perfiles de usuarios”, “obtención de recursos”, “modificación datos de la wiki”, etcétera.

El “grado” de interactividad como información de contexto fue observado específicamente durante la experiencia de taller expuesta en el capítulo IV bajo el título “Un diseño de taller físico-virtual-interactivo-comunicacional”. Retomando esta temática y atendiendo a la posibilidad de usar la información de interactividad como parámetro context-aware de los contratos según lo referido en “Elementos de la componente contrato”, de este mismo capítulo, podremos establecer un lazo de retroalimentación entre las prácticas efectuadas en la plataforma Moodle, establecidas en el registro de actividades y las acciones que devengan de los contratos. En esta sección utilizaremos un modelo particular de métrica y una propuesta de integración de dicho modelo al framework del dispositivo hipermedial dinámico a través de los contratos context-aware.

#### 5.5.1. MCondicionales

Para atender a los requerimientos anteriores, en primer lugar contamos con la aplicación Moodle con el agregado de la componente adicional contrato. El contrato fue incorporado a la herramienta foro de Moodle, mediante un prototipo experimental siguiendo el modelo arquitectónico desarrollado en el capítulo ??.

Cabe destacar que, si bien lo fundamentado sobre el agregado de la componente contrato a la arquitectura original de la plataforma está referido al proyecto Sakai, la misma es similar a la que proponemos para Moodle. En segundo lugar, diseñamos un mecanismo de métrica que fue usado como parte esencial de las reglas de los contratos, tomando como referencia un modelo de estandarización bien definido. Particularmente, la métrica formará parte del condicional de las reglas del contrato. A estos tipos de condicionales los denominaremos: “Mcondicional”. Conceptualmente no existe diferencia entre un Mcondicional y una condición común, los valores de verdad de ambos pueden depender de la invocación de métodos

## 5.5. Condicionales especiales para las reglas de los contratos

---

o funciones programadas para el sistema y colaborarán en la articulación de la ejecución de acciones (servicios y procesos computacionales).

Al incorporar las métricas en el modelo original de contratos contextaware, proponemos que el sistema sea más adaptable a nuevos cambios del contexto de usuario (por ejemplo, en concordancia con el tipo de interacción de la herramienta foro). Esto implica que las reglas que se pueden establecer para el contrato se formulan en el marco de la complejidad de los procesos didácticos (o investigativos y/o productivos), entendiendo que los mismos siempre exceden la posibilidad de ser absolutamente reglados. Por lo tanto, las mencionadas reglas estarán singularizadas en función de la dinámica de dichos procesos, debiendo ser explícitas y de clara implementación y por sobre todo, adaptables para los contextos específicos de los usuarios (en este caso, estudiantes a nivel de grado).

El primer paso es lograr la explicitación de las reglas del contrato y que los condicionales representen criterios de decisiones sobre aspectos relevantes del proceso didáctico (investigativo o productivo). Por ejemplo, un estudiante puede adquirir un servicio determinado de una herramienta, previo a la evaluación de una condición representada como condicional de una regla. En general, a estas reglas debemos diseñarlas con el cuidado de no incorporar redundancias, ambigüedades o incoherencias, tanto entre las propias reglas de un contrato como con otras reglas implícitas que se desprenden de los servicios. De esta manera, definimos las reglas del contrato como un conjunto de condiciones, acciones y prioridades.

La condición es una expresión Boolean sobre relaciones (mayor, menor, igual, distinto, etc.) entre parámetros y valores concretos. Las acciones conforman un conjunto de asignaciones de valor a otros parámetros también definidos por el tipo de regla. Algunos de los parámetros de las acciones deben ser “métodos de cálculo” que permitan cambios en el comportamiento de los servicios en los cuales estas reglas son aplicadas. La prioridad permite simplificar la cantidad de reglas que se deben escribir: en lugar de la escritura de una regla para cada combinación de posibilidades de los valores de los parámetros, se asegura que dos reglas no puedan ser ejecutadas simultáneamente. El usuario podría escribir una prioridad baja para todas las reglas y luego, con prioridades altas, ir identificando las excepciones para el caso configurado inicialmente.

En síntesis, las reglas son ejecutadas mediante un orden de prioridades. En consecuencia, en cada tipo de regla, cada una tiene sólo una prioridad. Para ejemplificar este concepto, consideremos una regla donde, dependiendo del tipo y cantidad de interacciones de un sujeto en formación (por ejemplo, Juan Pérez), el servicio de la herramienta foro, es readaptado. Cabe aclarar que en este caso, la acción del contrato establece el valor de calificación de la intervención del usuario Juan Pérez en los temas tratados en los foros. Entonces, el valor de verdad (Boolean) del condicional (donde parte de la expresión está representada por el predicado: `getForumTheme_Minimus_Forum_Theme`) *depende de la ejecución de una métrica, debido a que el método `getForum`*

la integración del subsistema de métrica como si fuera un sistema externo conectado por la componente contrato.

```
If (getStudent = \"Juan\") and (getForum_theme > Minimus_Forum_Theme)
Then
setPermissionQualifyForum (\"Juan\", Max_Forum_Qualify)
```

Ejemplo: Reglas de contratos con Mcondicionales

### Modelo de integración

Para implementar la invocación de métricas mediante métodos correctos, propusimos desde la perspectiva del rediseño e implementación computacional, un modelo de integración de muy bajo costo, sin cambios sustanciales ni en la arquitectura original ni en el código de la implementación.

El modelo conceptual de métrica pertenece al Modelo INCAMI (Information Need, Concept model, Attribute, Metric and Indicator: Información relevante, Modelo Conceptual, Atributos, Métricas e Indicadores) (Olsina, L., G. Lafuente y G. Rossi, 2000). INCAMI es un framework organizacional, orientado a la medición y evaluación que permite economizar consistentemente, no sólo metadata de métricas e indicadores, sino también valores mensurables en contextos físicos.

Por medio de un diagrama UML, se representa un modelo global de integración, teniendo en cuenta experiencias vinculadas al agregado de nuevas componentes en determinadas implementaciones resueltas para sistemas elearning similares Sakai. La integración contempla, por un lado, el modelo de coordinación de contrato (enmarcado en el área de contrato, y por el otro, el modelo de métrica propuesto por Olsina (Olsina, L. y G. Rossi, 2002). En la figura 3 podemos observar lo correspondiente a cada una de las áreas mencionadas. La figura 3 también nos muestra que la principal componente para lograr la integración está representada por la incorporación de una relación de agregación entre la componente contrato, representada por una clase en el modelo UML y la entidad método. Los condicionales de las reglas de los contratos son invocados (mediante un método explícito relacionado con la noción de los Mcondicionales, por ejemplo, getForumTheme) por medio de un mecanismo de callback que permite la correcta invocación de la métrica. El modelo de métrica proporciona una implementación y diseño de adecuadas métricas para

suplir los requerimientos de interacción referidos al registro de actividades. Si tenemos en cuenta el proceso de definición de métricas, debemos crear métricas por cada atributo. Definido en una estructura denominada árbol de requerimiento (tema que desarrollaremos más adelante), cada atributo puede ser cuantificado por más de una métrica. Cabe aclarar que este proceso de confección de métricas, podemos considerarlo como una metodología propia con fines específicos de esta área disciplinar de la ciencia. Las métricas contienen la definición de métodos y escalas para un determinado tipo de medición y/o cálculo. Dada una métrica  $m$  representa una relación (mapeo)  $m: A \rightarrow X$ , donde  $A$  es un atributo empírico de una entidad (el mundo empírico),  $X$  es la variable en la cual se asignarán (o pueden ser



## 5.5. Condicionales especiales para las reglas de los contratos

---

asignados) valores categóricos y numéricos (el mundo formal), y la flecha denota la relación de mapeo (mapping). Para lograr el mapping, es necesaria una correcta y precisa definición de actividades de medición por medio de una especificación explícita de los métodos de las métricas y las escalas tal como podemos ver en la figura ??.

Para las métricas directas, podemos aplicar métodos de mediciones objetivos y subjetivos, en cambio las métricas indirectas sólo pueden ser calculadas mediante funciones, con la intervención de fórmulas lógicas/matemáticas.

### 5.5.2. DMCondicionales

Como ya observamos, existen diferentes alternativas para lograr la integración de modelos por medio de un contrato. En este sentido, planteamos en la sección anterior un modelo de integración donde, por medio de los condicionales de las reglas, se invocaban métodos de un modelo externo. Ahora proponemos una nueva integración de un modelo externo que permitirá, aplicando la minería de datos, enriquecer aún más la semántica de los contratos.

Estado actual de la aplicación de la minería de datos a los sistemas de enseñanza mediados por la web En este apartado nos introduciremos en aspectos relevantes del estado del arte en la investigación sobre la aplicación específica de técnicas de minería de datos a los sistemas de enseñanza mediados por la web o sistemas e-learning. Tanto la minería de datos como dichos sistemas son áreas que muestran importantes desarrollos, y la vinculación de las mismas está suscitando un creciente interés por parte de investigadores y empresas.

Entre los diferentes métodos y técnicas existentes de minería de datos, nos hemos centrado principalmente en la minería de utilización web, concretamente en la clasificación y agrupamiento, descubrimiento de reglas de asociación y secuencias de patrones, ya que son técnicas motivadas por los mismos requerimientos planteados en este capítulo y fundamentados en la perspectiva de “Obra abierta”.

Asumimos que en los sistemas e-learning el principal objetivo de la utilización de técnicas de data mining (minería de datos) es proveer mecanismos que permitan guiar a los estudiantes durante su aprendizaje, con el propósito de incrementar las facilidades y servicios para la obtención de información y conocimiento calificado.

Las técnicas más utilizadas en la minería de datos aplicada a los sistemas de e-learning son: clasificación y agrupamiento, descubrimiento de reglas de asociación y análisis de secuencias. A continuación, vamos a exponer brevemente los principales trabajos de investigación agrupados dentro de estos tres tipos de técnicas, aunque algunos investigadores utilicen no una, sino varias. Clasificación y agrupamiento.

Las técnicas de clasificación y agrupamiento o clustering (Arabie, P., J. Hubert

y G. de Soete, 1996) consisten en la habilidad intelectual para ordenar o dividir fenómenos complejos (descritos por conjuntos de objetos con datos altamente dimensionales) en pequeños y comprensibles unidades o clases que permiten un mejor control o comprensión de la información.

Su aplicación a sistemas e-learning permite agrupar a los usuarios por su comportamiento de navegación, agrupar las páginas por su contenido, tipo o acceso y agrupar similares comportamientos de navegación. A continuación describiremos algunos trabajos de aplicación de minería de datos en educación.

La técnicas de agrupamiento son utilizadas por Gord McCalla y Tiffany Tang (Tang, T. y G. McCalla, 2004) para formar clusters o grupos de usuarios basándose en su comportamiento de navegación, aplicando un algoritmo de clustering basado en largas secuencias generalizadas. También proponen incluir un sistema recomendador inteligente dentro de un sistema de aprendizaje basado en web evolutivo capaz de adaptarse no sólo a sus usuarios, sino también a la web abierta. El sistema puede encontrar contenidos relevantes en la web pudiendo personalizar y adaptar sus contenidos, basándose en observaciones del sistema y por las propias valoraciones acumuladas dadas por los miembros en formación.

Otro trabajo que también emplea agrupamiento es el realizado por Elena Gaudioso y Luis Talavera (Romero, C., 2003) en el que analizan los datos obtenidos de cursos basados en sistemas e-learning y utilizan técnicas de clustering similares al modelo probabilístico de Naive Bayes para descubrir patrones que reflejan comportamientos de los usuarios. El objetivo es utilizar la minería de datos para dar soporte a la tutoría en comunidades de aprendizaje virtual.

La utilización conjunta de clustering con otras técnicas como secuenciación es realizada por Julia Miguillón y Enric Mor (Mor, E. y J. Minguillón, 2004) para analizar el comportamiento de navegación de los usuarios para la personalización del e-learning. Utilizan clustering de estudiantes para intentar extender las capacidades de secuenciación de algunos sistemas estándares de manejo de aprendizaje como SCORM para incluir el concepto de itinerario recomendado.

Los autores Erkki Sutinen y otros (Sutinen, E., W. Hämmäläinen, J. Suhonen y H. Toivonen, 2004) proponen un modelo híbrido que combina técnicas de minería de datos y de aprendizaje de máquinas para la construcción de una red bayesiana ([http://es.wikipedia.org/wiki/Red\\_bayesiana](http://es.wikipedia.org/wiki/Red_bayesiana)) que describe el proceso de aprendizaje de los estudiantes. Su objetivo es clasificar a los estudiantes para poder ofrecerles diferentes guías, dependiendo de sus habilidades y otras características. Esta tarea se realiza con la categorización y clustering de los estudiantes en base a sus habilidades o conocimiento. Finalmente, el trabajo realizado por Jing Luan (Romero, C., 2003) utiliza técnicas de minería de datos en educación superior y propone la utilización conjunta de predicción y clustering dentro de una herramienta de soporte de decisiones permitiendo a la universidad prever las necesidades de los estudiantes.

Antecedentes de técnicas de minería de datos Las principales aplicaciones de

## 5.5. Condicionales especiales para las reglas de los contratos

---

las técnicas de minería de datos en educación son sistemas de personalización (Srivastava, J., B. Mobasher y R. Cooley, 2000), sistemas recomendadores (Li, J. y O. Zaiane 2004), sistemas de modificación (Perkowitz, M. y O. Etzioni, 1998), sistemas de detección de irregularidades (Barnett, V. y T. Lewis, 1994), etcétera, debido a sus capacidades para el descubrimiento de patrones de navegación regulares e irregulares, clasificaciones de alumnos y de los contenidos, construcción adaptativa de planes de enseñanza, descubrimiento de relaciones entre actividades, diagnóstico de los estudiantes, etcétera.

Reglas de los contratos y los DMcondicional El uso de reglas es una de las formas más usuales de representación del conocimiento debido, entre otras razones, a su sencillez, capacidad de expresión y escalabilidad. Dependiendo de la naturaleza del conocimiento que almacenan, se ha establecido una tipología informal para este tipo de estructuras. Así, se habla de reglas de decisión, asociación, clasificación, predicción, causalidad, optimización, etc. En el ámbito de la extracción de conocimiento en bases de datos, las más estudiadas han sido las reglas de asociación, de clasificación y de predicción.

Las reglas de clasificación tienen como objetivo almacenar conocimiento encaminado a la construcción de un clasificador preciso. En su antecedente contienen una serie de requisitos (en forma de condiciones) que debe cumplir un objeto determinado para que pueda considerarse perteneciente a la clase identificada con el consecuente de la regla. Desde el punto de vista sintáctico, la principal diferencia con las reglas de asociación es que presentan una sola condición en el consecuente, que además pertenece a un identificador de clase.

La estructura de una regla puede ser caracterizada (en términos generales) mediante un conjunto de condicionales lógicos en los que se fija un criterio de ejecución de la acción. En este ejemplo, se verifica en el usuario "alumno", si su nivel de interacción en el foro pertenece a la clase de alto, y si la calidad de sus intervenciones se encuentran en un rango aceptable. Este tipo de información es obtenida del registro de actividades de la plataforma.

Supongamos, ahora, que los valores de estos dos últimos condicionales se obtienen a partir de la ejecución de un algoritmo tipo TDIDT (Top-Down Induction of Decision Trees), entonces, debe ser invocado el algoritmo por medio de un método (en este caso, `getCalidadIntervencion`) a través de una regla explícita en el contrato, de la siguiente manera:

```
if (usuario = 'alumno') and getCalidad_Intervencion = 'aceptable')
and this.getCantidad_Interaccion='alto')
then herramienta.accion ()
```

Interesa destacar la estructura de la regla Si-entonces (If-Then) y los condicio-

## 5.5. Condicionales especiales para las reglas de los contratos

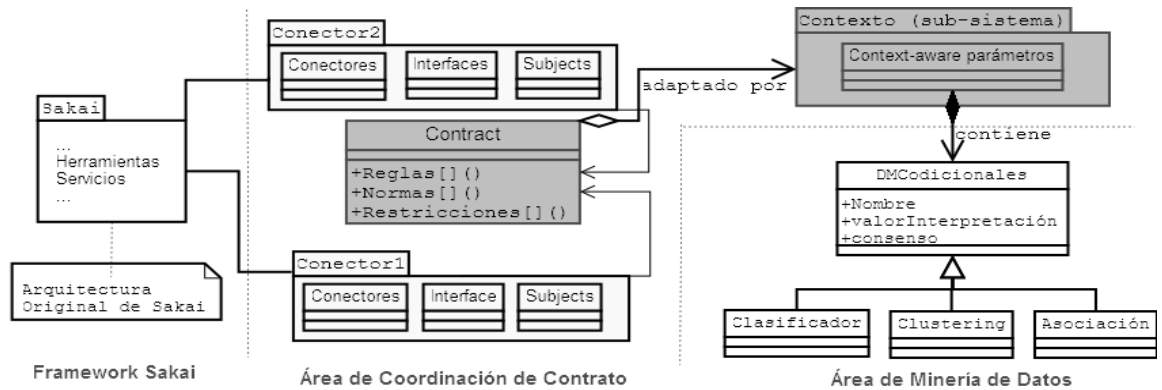


Figura 5.4: Modelo de integración de los MDCondicionales

nales de los cuales se desprende la ejecución de un algoritmo de data mining. En este ejemplo, este tipo de condicional se muestra subrayado; y lo denominaremos *MDcondicional* (en lo conceptual, tenemos gran similitud con los *Mcondicionales* de la sección “Métricas de interacción para ‘Obra abierta’”). De esta manera, por medio del *MDcondicional*, se capturan los resultados de la inferencia del árbol de decisión que se ilustra con la figura ??.

En las secciones posteriores nos introduciremos en las propuestas y justificaciones para la incorporación de las técnicas de minería de datos a través de los *MDcondicional*.

**Algoritmos de construcción de los *MDcondicionales*** Una de las formas más simples de representar conocimiento es mediante árboles de decisión. Dicha estructura de representación del conocimiento está formada por una serie de nodos, donde: cada nodo interno es etiquetado con el nombre de uno de los atributos predictores; las ramas que salen de un nodo interno son etiquetadas con los valores del atributo de ese nodo; cada nodo terminal, o nodo hoja, es etiquetado con el valor del atributo objetivo.

Un ejemplo de árbol de decisión se muestra en la figura 4, donde el atributo objetivo es el atributo nivel de aceptación y los atributos predictores son los atributos: interacción, respuestas, evaluación, consultas. Una característica muy interesante de los árboles de decisión es que, a partir de ellos, es muy fácil derivar un conjunto de reglas de producción del tipo Si-entonces (If-Then) completamente equivalente al árbol original. El algoritmo que nos permite realizar este cambio de modelo de representación es casi trivial y convierte cada camino que va desde el nodo raíz hasta una hoja, en una regla de la siguiente forma: los nodos internos y sus correspondientes ramas de salida son convertidos en las condiciones del antecedente de la regla; los nodos hojas se convierten en el consecuente de la regla. Al convertir el árbol de decisión en una colección de reglas se obtiene una regla por hoja del árbol. Dichas reglas serán, por tanto, mutuamente excluyentes.

El algoritmo de construcción de árboles de decisión más popular es el algoritmo ID3 (Quinlan, J. R., 1987). Este algoritmo está basado en la división sucesiva y

## 5.5. Condicionales especiales para las reglas de los contratos

---

construye el árbol de decisión desde la raíz hacia las hojas, incrementando en cada paso la complejidad del árbol.

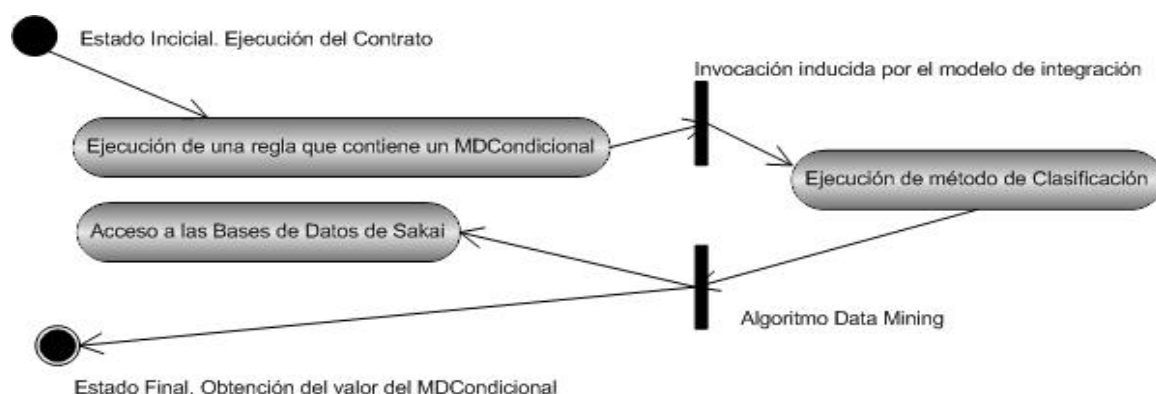
Inicializar T con el conjunto de todas las instancias Si (todas las instancias en el conjunto T satisfacen el criterio de parada) Entonces Crear un nodo hoja etiquetado con el nombre de la clase y parar.

Sino Seleccionar un atributo para utilizar como atributo de particionado. Crear un nodo etiquetado con el nombre del atributo y crear una rama por cada valor del atributo. Particionar T en subconjuntos tales que contengan las instancias que cumplan los valores del atributo.

Aplicar este algoritmo recursivamente para cada subconjunto. Fin Si Recorrer el árbol y formar las reglas.

### Detalles del modelo de integración

Para lograr la incorporación de la componente data mining, volvemos a proponer una nueva integración a muy bajo costo, desde el punto de vista del rediseño e implementación con el propósito de facilitar la misma y el impacto en la modificación de los módulos estándares de la aplicación e-learning. En particular, Sakai, a diferencia de Moodle, provee una arquitectura en la que es posible trabajar bajo el paradigma orientado a objetos (POO), con toda la potencia y versatilidad de la tecnología Java, motivo por el cual se logra una mejor adaptación del modelo de la componente conceptual data mining mediante un correcto diseño orientado a objeto (DOO) (Grady, B., 1982). En la figura 5, representamos el contrato con un diagrama de clase. Las reglas, normas y restricciones (RNR) están representadas con tres métodos independientes en el diagrama de la clase contrato. Teniendo en cuenta la analogía planteada entre las reglas del contrato y la estructura de los árboles que determina el algoritmo TDIDT, es posible plantear un nuevo mecanismo total o parcial de construcción de reglas dentro de los contratos. Esta nueva perspectiva puede ser conceptualizada de las siguientes formas: 1) el uso de técnicas de data mining permite una construcción automática de reglas. Bajo los contratos (representado en la figura 5 como una relación de agregación en UML) se realiza dicha construcción en tiempo de ejecución. En este caso se aporta automatización, adaptabilidad y dinamismo; 2) proporciona una manera indirecta de recolección de información de contexto, debido a la naturaleza de este tipo de técnicas (minería de datos) donde la fuente de información procesada reviste a datos empíricos y valores que denotan tipos de relaciones. Con esta información de contexto, los contratos adquieren mayores niveles expresivos en cuanto a su sensibilidad al contexto, similar a los sistemas context-aware.



Cuando, dentro en una NRR se utiliza un método para implementar algunas de las técnicas de data mining (en este caso, la construcción de un árbol de decisión por medio de un algoritmo TDIDT), queda conformada una regla donde el valor de su condicional (el cual fue denominado MDcondicional) responde –conceptualmente– a una estructura, en este caso: un árbol. El vínculo entre las reglas del contrato y la efectiva representación del TDIDT se concreta a través de una relación de agregación entre los parámetros del contrato (representado por la clase “parámetros context-aware”, que se encuentra dentro del subsistema que engloba el contexto, similar a la propuesta de Schmidt, A., 2005) y la clase DMcondicionales (representada como una generalización de subclases donde se distinguen las posibles técnicas de data mining).

En la figura ??, las componentes significativas para la integración de los modelos se encuentran enmarcadas en color gris. La primera pertenece al área de coordinación de contrato y mantiene la estructura original del modelo pro-

puesto en el proyecto “Obra abierta”. La segunda componente se encuentra representada por una clase conceptual que simboliza las características y funcionalidades de minería de datos, articuladas para cumplir los objetivos y requerimientos propuestos.

**Mecanismo de comunicación** La figura 5 es una reelaboración significativa de integración arquitectónica entre dos modelos (el framework e-learning y el modelo de minería de datos) donde se muestran, además, relaciones salientes de dependencias entre objetos de las distintas áreas. Por ejemplo, cuando un objeto perteneciente a una herramienta5 de la aplicación atiende un pedido de un usuario, se comunica con otro objeto (en este caso un objeto contrato, perteneciente al área de coordinación de contrato) con el propósito de verificar la existencia de una regla que determine cuáles son las acciones que se deben tomar, teniendo en cuenta determinadas condiciones. Si alguna de las condiciones está representada por un MDcondicional, entonces existirá una comunicación entre el objeto contrato (que contiene la regla) y un objeto del área de minería de datos. Esta cadena de eventos y relaciones entre áreas permite que un servicio ordinario del framework Sakai se pueda enriquecer con información de contexto recopilada con técnicas de minería de datos.

## 5.5. Condicionales especiales para las reglas de los contratos

---

**Caso de uso** Retomamos un requerimiento similar al propuesto en la sección “Métricas de interacción para ‘Obra abierta’ ”, y agregamos que el análisis de información almacenada es utilizada como recompilación de información (datos) de contexto.

En este ejemplo, cambiaremos pequeños aspectos de los requerimientos originales para poder involucrar las técnicas de minería de datos como parte de la implementación, refiriendo el caso sobre el entorno Sakai. Supongamos ahora que volvemos a la hipótesis de contar con un dispositivo hipermedial conformado con la arquitectura del modelo de integración y además, la implementación del modelo de la figura 6. Sabemos que la herramienta se encuentra configurada con un contrato (del tipo de la sección “Los contratos context-aware”) conformado, a su vez, con una regla similar a la presentada en la sección “Modelo de integración”.

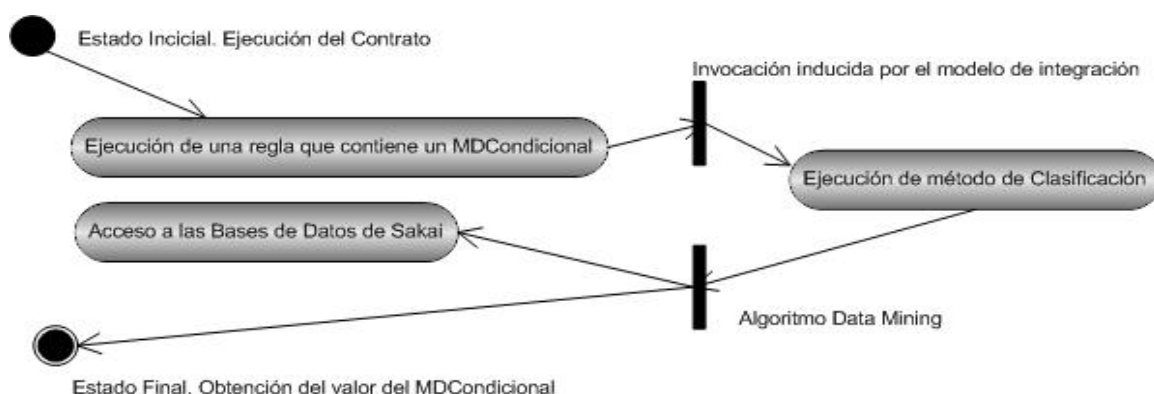
Entonces, cuando un usuario alumno ingrese al foro y ejecute cualquier tipo de servicios donde intervenga el contrato anterior, las prestaciones que recibirá estarán supeditadas a los valores que se obtengan de la ejecución del clasificador en base a la información obtenida a partir del registro de actividades de la plataforma Sakai. Cabe aclarar que Sakai contiene un registro de actividades al cual podemos acceder desde las tablas de las bases de datos. La implementación y operatoria que implique la clasificación de los datos desde los *MDcondicionales* es responsabilidad del subsistema que lo implementa, respetando de esta manera la propiedad de independencia entre los modelos de la integración señalada en la sección “Detalles del modelo de integración”.

Si profundizamos en el tipo de relación que se establece a través de los *MDcondicionales* y tomamos como referencia el punto de vista del usuario, podemos caracterizar el flujo de ejecución de órdenes de la siguiente manera: en la figura 6, podríamos considerar un estado inicial, en el que el sistema ejecuta las reglas de un contrato que contiene una de ellas con un *MDcondicional*. Por medio de un método, se produce la invocación del método de clasificación que se encuentra implementado en el subsistema de minería de datos (referenciado con la entidad “modelo externo” en la sección “El contrato como conector”).

Las transacciones y estados que se suscitan son luego transparentes para el modelo de integración y pueden resumirse en la ejecución de un algoritmo de data mining donde se provoca una consulta a la base de datos Sakai y la ejecución de métodos de clasificación (similares a los de la sección “Antecedentes de técnicas de minería de datos”) para obtener el valor de verdad del *MDcondicional*, llegando, así, al estado final del ciclo de ejecución.

Con este tipo de configuración, la herramienta foro de Sakai adquiere características context-aware y el agregado de la siguiente propiedad: los contratos context-aware pueden ser modificados en tiempo de ejecución. Esto quiere decir, que el mismo alumno, ante una nueva interacción con el foro y bajo las mismas condiciones, puede obtener resultados diferentes en el caso de que se hayan modificado las reglas del contrato. Además, la incorporación de un subsistema para la aplicación de técnicas de minería de datos ofrece mayor potencial expresivo para la construcción de reglas en los contratos.

Finalmente, sobre esta temática consideramos que el uso de un correcto modelo para la implementación de técnicas de minería en la clasificación de valores de contextos permite potenciar las soluciones creadas para los dispositivos hipermediales dinámicos para educación e investigación. Es importante entonces, que el grupo de diseñadores, expertos y desarrolladores sean competentes en la utilización comprensiva de este tipo de técnicas.



### Hacia la implementación

Con el objetivo de profundizar la visión sobre el diseño de dispositivos hipermediales dinámicos, abordamos en este capítulo especificaciones técnicas que justificaron la implementación y diseño de modelos de integración donde la componente contrato adoptaba el rol de conector. Observamos, entonces, que la base conceptual para que el contrato pueda ser visto y manipulado en su diseño e implementación como una pieza de software, se centra en el rol protagonista y responsable que debemos adoptar como expertos diseñadores de dispositivos hipermediales dinámicos en el ciclo de vida del mismo (diseño, implementación, uso).

Es a partir de esta nueva toma de posición, que la etapa de diseño adquiere una dimensión activa centrada en la participación con la puesta en obra de la teoría de coordinación de contratos context-aware.

Tomamos conciencia de que el camino hacia la implementación del marco teórico propuesto demanda detenimiento para su comprensión, integración de conocimientos de distintas disciplinas, diálogo interdisciplinar para poder alcanzar claros criterios en la utilización del contrato y en la explicitación de las reglas singulares que lo componen, configuradas en función de los procesos educativos, investigativos o de producción que las susciten. Teniendo en cuenta su poder expresivo, podremos desarrollar adecuadas aplicaciones contextualizadas que nos abran la posibilidad de desarrollar nuestra formación continua y creatividad, construyendo polifónicamente una singular mesa de arena.



## **5.6. Conclusiones**

---

## Framework para la inyección de contratos sensibles al contexto en entornos Web colaborativos

---

Environment

### 6.1. Introducción

A medida que el avance en la investigación y desarrollo de plataformas e-learning brindan mejoras e innovaciones en herramientas (videoconferencias, portfolios, wikis, workshops, etc.) y sus respectivos servicios, crece la cantidad de posibles configuraciones de los espacios e-learning. Estas configuraciones abarcan diferentes tipos de requerimientos pertenecientes a las etapas de diseño, desarrollo e incluso exigen que el espacio e-learning se adapte en tiempo de ejecución. A partir de estos requerimientos se definen los procesos e-learning (que nosotros denominamos Pe-Irn) (67) de manera semejante a procesos de negocio en otro dominio de aplicación. Al igual que los procesos de negocios en una Aplicación Web convencional, los Pe-Irn están compuestos por transacciones Web (). En este contexto, una transacción (o transacción e-learning) es definida como una secuencia de actividades que un usuario ejecuta a través de una Aplicación e-learning con el propósito de efectuar una tarea o concretar un objetivo, donde el conjunto de actividades, sus propiedades y las reglas que controlan sus ejecuciones dependen del Pe-Irn que la Aplicación debe brindar. Un ejemplo de estrategia didáctica es la posibilidad de que un alumno acceda a determinado tipo de material (vídeos,

## 6.1. Introducción

---

archivos, etc.) dependiendo de sus intervenciones en los Foros. Estos tipos de requerimientos resultan difíciles de implementar con las actuales aplicaciones e-learning de extendido uso a nivel global.

Las características funcionales de las actuales aplicaciones Web e-learning (AWe-Irn), como Sakai<sup>a</sup>, se basan en brindar navegación entre páginas a través de links y ejecución de transacciones e-learning por desde las herramientas (ej., Foros, Anuncios, Exámenes, Blogs, etc.) que utilizan los servicios de la plataforma (ej., edición, manejo de audio y vídeo, consultas, navegación, etc.). y la forma en que sus configuraciones pueden ser alteradas, son actualmente un fuerte impedimento para implementar requisitos como el mencionado anteriormente. tecnológicos involucrados en las etapas de diseño y, principalmente, en la etapa de desarrollo e implementación.

En el marco de los análisis efectuados podemos sostener que el proyecto Sakai brinda una de las propuestas más consolidadas de diseño y desarrollo de entornos colaborativos e-learning para educación, orientado a herramientas que se implementan a través de servicios comunes (servicios bases). Por ejemplo, el servicio de edición de mensajes es utilizado en las herramientas Foro, Anuncio, Blog, PorFolio, etc. Más aun, otras de las características salientes de Sakai es la versatilidad para su extensión y/o configuración. En efecto, Sakai permite alterar ciertas configuraciones en tiempo de ejecución, por ejemplo, instrumentar una nueva funcionalidad en un servicio base de Sakai.

Sin embargo, estas soluciones no pueden resolver aquellos Pe-Irn que involucren cambios en el comportamiento de la relaciones entre un componente (cliente) que ocasiona, a través de un pedido, la ejecución de un componente servidor (proveedor). Estos tipos de cambios refieren a la capacidad de adaptación dinámica del sistema (73) extendiendo, personalizando y mejorando los servicios sin la necesidad de recompilar y/o reiniciar el sistema. En este trabajo se presenta un propuesta para la incorporación (agregado) de propiedades de adaptación dinámicas a los servicios bases del framework Sakai, especialmente diseñadas para implementar Pe-Irn definidos en el marco de (69) donde se requieren nuevos aspectos de adaptación (57) en la perspectiva de los Dispositivos Hipermediales Dinámicos en el campo del e-learning (71) con característica *context-aware* (58? ). Se entiende por DHD *“a una red social mediada por las TIC en un contexto físico-virtual-presencial, donde los sujetos investigan, enseñan, aprenden, dialogan, confrontan, evalúan, producen y realizan responsablemente procesos de transformación sobre objetos, regulados según el caso, por una coordinación de contratos integrados a la modalidad participativa del Taller”*. (San Martín et al, 2008)

El resto de este trabajo se encuentra organizado de la siguiente manera: Tras esta introducción se presenta una referencia conceptual del framework contratos con características “context-aware“, utilizados en el proyecto Obra Abierta: Dispositivos Hipermediales Dinámicos para educar e investigar (Dir. Dra. Patricia S. San Martín (Sección 6.2). Luego se describe la macro-arquitectura de integración

---

<sup>a</sup>Sakai: Entorno colaborativo y de aprendizaje para enseñar. Es de código abierto y está resuelto con tecnología Java. Véase detalles en [www.sakaiprojet.org](http://www.sakaiprojet.org)

entre los frameworks de Sakai, la teoría de coordinación de contratos sensible al contexto (TCCs-c), un modelo de contratos context-aware y aplicaciones externas (sección 6.3). Siguiendo, se presenta un modelo de implementación de la TCCs-c a través del de patrones de diseño (sección 6.4). En la sección 6.5 se propone un caso de estudio concreto. Finalizando con una breve conclusión.

## 6.2. Contratos con características sensibles al contexto

Nuestra propuesta de solución a los requerimientos mencionados sobre adaptación dinámica comienza con la construcción de un modelo de contrato orientado a la implementación de servicios sensibles al contexto. El uso de contratos parte de la noción de Programación por Contrato ("Programming by Contract") de Meyer (70) basada en la metáfora de que un elemento de un sistema de software colabora con otro, manteniendo obligaciones y beneficios mutuos. En nuestro dominio de aplicación consideraremos que un objeto cliente y un objeto servidor "acuerdan" a través de un contrato (representado con un nuevo objeto) que el objeto servidor satisfaga el pedido del cliente, y al mismo tiempo el cliente cumpla con las condiciones impuestas por el proveedor. Como ejemplo de la aplicación de la idea de Meyer en nuestro dominio de sistemas e-learning planteamos la situación en que un usuario (cliente) utiliza un servicio de edición de mensajes (servidor) a través de un contrato que garantizará las siguientes condiciones: el usuario debe poder editar aquellos mensajes que tiene autorización según su perfil (obligación del proveedor y beneficio del cliente); el proveedor debe tener acceso a la información del perfil del usuario (obligación del cliente y beneficio del proveedor).

A partir de la conceptualización de contratos según Meyer se propone una extensión por medio del agregado de nuevas componentes para instrumentar mecanismos que permitan ejecutar acciones dependiendo del contexto.

En aplicaciones sensibles al contexto (66), el contexto (o información de contexto) es definido como la información que puede ser usada para caracterizar la situación de una entidad más allá de los atributos que la definen. En nuestro caso, una entidad es un usuario (alumno, docente, etc.), lugar (aula, biblioteca, sala de consulta, etc.), recurso (impresora, fax, etc.), u objeto (examen, trabajo práctico, etc.) que se comunica con otra entidad a través del contrato. En (58) se propone una especificación del concepto de contexto partiendo de las consideraciones de Dourish (79) y adaptadas al dominio e-learning, que será la que consideraremos en este trabajo. Contexto es todo tipo de información que pueda ser censada y procesada, a través de la aplicación e-learning, que caracterizan a un usuario o entorno, por ejemplo: intervenciones en los foros, promedios de notas, habilidades, niveles de conocimientos, máquinas (direcciones ip) conectas, nivel de intervención en los foros, cantidad de usuarios conectados, fechas y horarios, estadísticas sobre cursos, etc.

## 6.2. Contratos con características sensibles al contexto

En términos generales, la coordinación de contratos es una conexión establecida entre un grupo de objetos aunque en este trabajo se consideran sólo dos objetos: un cliente y un servidor.

Cuando un objeto, cliente efectúa una llamada a un objeto servidor (ej., el servicio de edición de la herramienta Foro), el contrato "intercepta" la llamada y establece una nueva relación teniendo en cuenta el contexto del objeto cliente, el del objeto servidor, e información relevante adquirida y representada como contexto del entorno (79). Como condición necesaria, el uso de contratos no debe alterar la funcionalidad de la implementación de los objetos participantes, aunque sí se espera que altere la funcionalidad del sistema.

A continuación se presenta un modelo conceptual resumido de contratos sensibles al contexto. Se brindarán detalles sobre algunas de los componentes y relaciones esenciales para la integración de este modelo con el framework Sakai y con los módulos que instrumentan la coordinación de contratos 6.3.

### 6.2.1. Elementos de los contratos sensibles al contexto

Un contrato que siga las ideas de Meyer contiene toda la información sobre los servicios que utilizarán los clientes. Para incorporar sensibilidad al contexto nuestros contratos deberán tener referencias sobre algún tipo de información de contexto para su utilización.

En el diagrama de relaciones entre entidades mostrado en la Figura 6.1 se describen los elementos que componen el concepto de contrato sensible al contexto, especialmente para Pe-Irn.

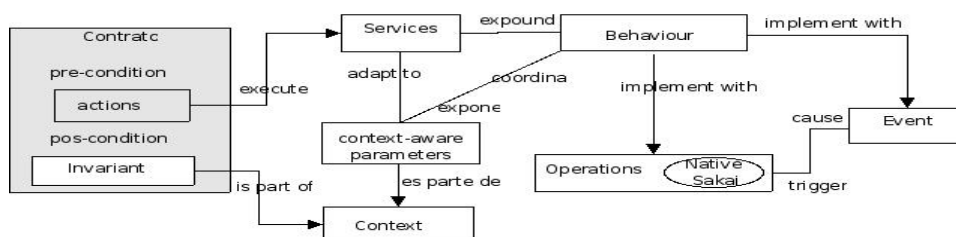


Figura 6.1: Modelo de un contrato context-aware

El propósito de la presentación de este modelo de contrato sensible al contexto es identificar cuales son los componentes que lo componen y que serán referenciados en la siguiente sección para la implementación de un modelo de integración con el framework Sakai.

La figura comienza con la representación de un contrato según Meyer donde se

caracterizan los principales elementos que lo componen (pre-condiciones, acciones, pos-condiciones). Las flechas salientes de la zona gris indican los dos tipos de relaciones que se debe instrumentar para incorporar un mecanismo que provea a los contratos la característica de sensibilidad al contexto. La primera de estas relaciones indica que desde las acciones del contrato se invocan a los servicios que provee un objeto servidor. La segunda relación se da debido los invariantes pueden depender de información de contexto. En la porción derecha de la Figura 6.1 aparecen las entidades necesarias para obtener contratos sensibles al contexto. Las explicamos a continuación.

**Servicios:** En esta componente se representan los elementos necesarios para la identificación de un servicios y clasificación de los servicios que pueden formar parte de las acciones de los contratos. Por ejemplo, nombre del servicio, identificadores, alcance, propósito, etc. Para más detalles consultar (? ). El comportamiento funcional de cada servicio se expone a través de la componente **Comportamiento**.

**Comportamiento:** El comportamiento de un servicio se logra a partir de combinar operaciones y eventos que son representadas con el las componente **Operaciones** y **Eventos**. De la misma manera el servicio puede ser implementado a través del uso de eventos, representados con el componente **Eventos**, que puede lanzado operaciones del componente **Operaciones**. Por ejemplo, de acuerdo con los roles (ej., alumno, instructor, docente, etc.) asignados a un usuario de una herramienta involucrado en un Pe-Irn, en un determinado contexto del entorno (ej., si está en un espacio Foro) y del usuario (ej., si tiene permiso de moderador), la componente **servicio** brindan distintas funcionalidades (ej., editar un mensaje), que son instrumentadas por medio de operaciones concretas (ej., guardar un mensaje en una tabla) y/o a través de la publicación o subscripción de eventos. Las operaciones pueden ser de dos tipos: operaciones que cambian el estado del sistema (tipo “update”) y/o operaciones que proveen algún tipo de información sobre consultas (tipo “query”). Estas operaciones pueden ser creadas creadas por el programador o utilizadas a partir de las operaciones provista por el núcleo del framework sakai (ej., adquirir el id de un canal de mensajes) usadas en los servicios bases (ej., el servicio de edición de mensajes).

**Parámetros Context-Aware:** Se denomina **parámetros context-aware** a la representación de la información de contexto que forma parte de los parámetros de entrada de las funciones y métodos exportados por los servicios, estableciendo de esta manera una relación entre el componente **Servicios** y el componente **Parámetros contex-aware**. La influencia de estos parámetros en el comportamiento funcional de los servicios es representada a través de la relación entre los componentes **Parámetros context-aware** y **Comportamiento**.

**Contexto:** Este componente representa el contexto o información de contexto definida anteriormente en esta sección. Para nuestro modelo este tipo de información es utilizada de dos maneras diferentes: en primer lugar para la asignación de los valores que toman los **Parámetros context-aware**; en segundo lugar esta información puede ser utilizada para definir los invariantes que se representan en los contratos.

## 6.3. Modelo de integración de Sakai con contratos

En esta sección se muestra un esbozo de la propuesta para incorporar a Sakai un mecanismo de coordinación de contratos sensibles al contexto, a través de la presentación de una diseño que describe la comunicación entre módulos (74) y sus dependencias. En la figura 6.2 los módulos son representados con paquetes UML y las relaciones entre ellos por medio de flechas que indican comunicación y dependencias. A su vez, en un segundo plano se muestra cuáles son las clases específicas de cada módulo y cómo se implementa la integración a través de estas.

El framework Sakai, representado en la figura 6.2 por el módulo Sakai, está diseñado según una arquitectura de cuatro capas (?): La capa de **agregación** se encarga completamente de la implementación de la interfaz con el usuario al estilo de las implementaciones de portales Web. La segunda capa denominada **presentación** tiene la responsabilidad de permitir la reutilización de los componentes Web (ej., "widget" que proveen calendarios, editores "WSYWIG", etc.). La tercera corresponde a la capa de **herramientas** donde reside la lógica de negocio de las herramientas Sakai que interactúan con el usuario (ej., Foro, Wiki, etc.). Por último la capa de **servicio** implementa los servicios Sakai bajo una Arquitectura Orientada a Servicios que serán utilizados por varias herramientas a través de API.

Para nuestra propuesta de integración tendremos en cuenta únicamente servicios que componen laa capa de **servicios** pertenecientes a los servicios bases Sakai. Los servicios del núcleo Sakai serán envueltos mediante la coordinación de contratos, lo que se representa en la Figura 6.2 con las referencias del **Módulo Sakai** hacia el **Módulo de Coordinación de Contratos**.

De esta forma, se logra controlar las invocaciones a los servicios del núcleo Sakai a través del **módulo Coordinación de Contratos**, quedando establecida una primer relación de integración. Esta misma relación es implementada a nivel de clases como una relación de agregación entre la clase *Servicios Sakai* (correspondiente al módulo Sakai) y la clase *Conector* correspondiente al framework de coordinación que será tratado en la sección 6.4 a través del uso de patrones de diseño.

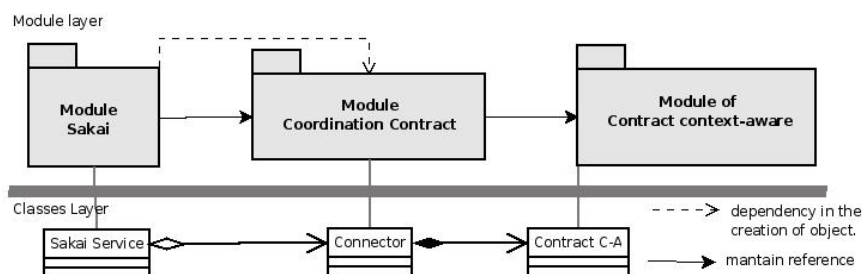


Figura 6.2: Diseño de la integración de Sakai con Contratos

La última relación de integración se produce entre el módulo de **Coordination**

**de Contratos** y el módulo de **Contratos Sensible al contexto** el cual fue desarrollado en la sección 6.2. En este caso, la figura 6.2 muestra cómo desde el framework de coordinación se establece una relación de composición entre la clase *Conector* y la clase *Contrato C-A* que implementa el elemento Contrato representado en la Figura 6.1 coloreada en gris.

De esta forma, instanciando dinámicamente diferentes contratos, Sakai presentará una flexibilidad en tiempo de ejecución que actualmente no posee y que es la que requieren los DHD.

## 6.4. Implementación de contratos en Sakai

En las secciones anteriores se mencionó al contrato como un agente activo que se encarga de la coordinación de las componentes que lo integran. En esta sección se describe cómo este mecanismo fue implementado a través de una herramienta que permite modificar implementaciones de clases Java de los servicios Sakai para incorporarle mecanismos de coordinación de contratos sensibles al contexto, según el diseño mostrado en la Figura 6.2. Sin embargo creemos que antes es conveniente resumir brevemente otros intentos de proveer la flexibilidad requerida por los DHD.

Lograr este grado de flexibilidad es absolutamente necesario teniendo en cuenta que la implementación de estas clases, para los frameworks colaborativos e-learning como Sakai, no pueden ser modificadas (rediseñadas, reemplazadas). Para estos casos, la implementación de la TCCs-c proporciona un mecanismo que se distingue de propuestas existentes para el modelado de interacción entre objetos.

### 6.4.1. Niveles de flexibilidad de Sakai

Diferentes estándares y frameworks de desarrollo de software basados en componentes e inyección de servicios fueron propuestos para mejorar la flexibilidad y adaptabilidad<sup>b</sup> de los sistemas e-learning Web, los más importantes son: CORBA, JavaBeans, Hibernate, Spring, JSF, RSF, etc. Sin embargo, ninguno de estos estándares proveen un mecanismo convenientemente implementativo y abstracto en donde se permita representar a las relaciones entre usuarios y servicios como un objeto de primera clase (70). En este sentido, se implementa una solución utilizando patrones de diseño que implemente características deseadas de la TCCs-c sobre la capa de servicios base de framework Sakai.

Desde un punto de vista implementativo, esta propuesta cambia la filosofía Java2EE-Sakai<sup>c</sup> donde la incorporación de nuevas prestaciones de versatilidad para

---

<sup>b</sup>La adaptabilidad en el sentido que se proponen en los sistemas hipermediales que permiten personalizarse (adaptarse) en función de los usuarios individuales (Henze, 2000).

<sup>c</sup>Sakai mantiene la filosofía de desarrollo de Java2, de aplicaciones orientadas a servicios



## 6.4. Implementación de contratos en Sakai

la configuración e implementación se logran a través de extensiones de clases, relaciones a librerías, técnicas de reflexión (Reflection"), etc.

En el siguiente diagrama de clases UML se muestra el diseño propuesto para incorporar coordinación de contratos sensibles al contexto en Sakai. A continuación explicamos cada uno de los componentes y sus relaciones.

### 6.4.2. Patrones de diseño para la coordinación de contratos sensible al contexto

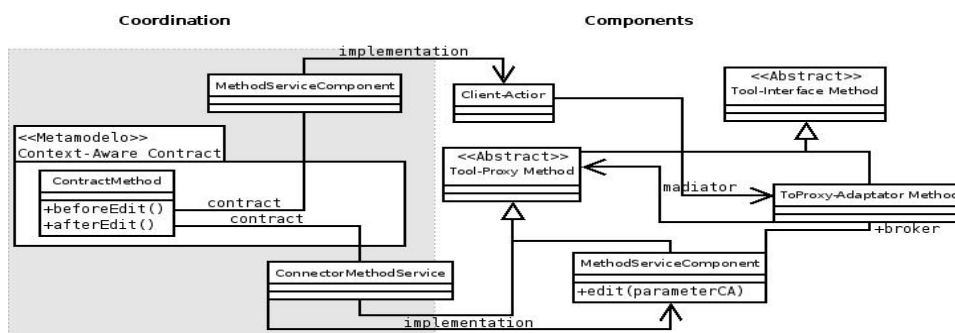


Figura 6.3: Diseño del modelo de integración

**MétodoHerramienta\_Interfaz:** Como su nombre lo indica, esta es una clase abstracta que define una interfaz común de los servicios de una Herramienta provisto por **MétodoHerramienta\_Proxy** y **MétodoToProxy\_Adaptador**

**MétodoToProxy\_Adaptador:** Esta es una clase concreta que implementa a un intermediario manteniendo la referencia a la clase *Método\_Proxy* para encargarse de los pedidos recibidos. En tiempo de ejecución, esta entidad puede indicar a **MétodoServiceComponente** que no hay contratos involucrados o instrumentar una conexión entre **ConectorMétodosService** para que conecte la clase real **MétodoServiceComponente** con el contrato **ContractMétodo** en el que se encuentran las reglas de coordinación.

**MétodoHerramienta\_Proxy:** Es una clase abstracta que define la interfaz que tienen en común **MétodoHerramientaToProxy\_Adaptador** y **ConectorMétodoService**. La interfaz es heredada de **MétodoHerramienta\_Interfaz** con el propósito de garantizar que ofrezca la misma interfaz de **MétodoHerramientaToProxy\_Adaptador** con la cual un cliente real debe interactuar (ej., un objeto que invoque el servicio de edición de un Foro).

**EditServiceComponente:** Es la clase concreta donde se encuentra la lógica de la edición del Foro y la implementación concreta de los servicios originales de Sakai.

("service-oriented") que pueden ser escalables, fiable, interoperable y extensible.

**ConectorMétodoService:** Efectúa la conexión entre el contrato y los objetos reales (en este caso **MétodoServiceComponente**) involucrados como partes del contrato. De esta manera, no se requiere la creación o instanciación de nuevos objetos para coordinar el mismo objeto real agregando o sacando otros contratos. Esto es posible solamente con una nueva asociación a un nuevo contrato y con la instanciación de una conexión con una instancia existente de **ConectorMétodoService**. Esto significa que hay una sola instancia de esta clase asociada con una instancia de **MétodoServiceComponente**.

**ContractMétodo:** Es la clase que cuya instancia genera un objeto de coordinación que al ser notificado toma decisiones siempre que un pedido es invocado a través de un objeto real. Para los casos en que no haya contratos coordinando un objeto real, el diseño de la figura 6.3 puede ser simplificado y sólo las clases y relaciones que no pasen por la zona del rectángulo gris son necesarias. La introducción de un contrato implica la creación de las clases y relaciones enmarcadas en la zona gris. La clase que implementa al contrato, llamada **ContractMétodo**, se encuentra representada dentro de la figura de un paquete UML, llamado **Contratos Sensible al Contexto**, indicando su pertenencia a un modelo subyacente descrito en la sección 6.2.

Este mecanismo de intercepción permite imponer otra funcionalidad en la interacción entre componentes del llamante (por medio de su pedido) y del llamado (a través de su respuesta). De esta manera, desde el aspecto tecnológico se instrumenta la posibilidad de brindar un nuevo “grado de libertad” a los servicios de Sakai, permitiendo establecer configuraciones propias de un DHD en tiempo de ejecución.

Por lo expuesto, es válido reforzar la argumentación sobre las ventajas de esta propuesta en comparación con las actuales soluciones tecnológicas basadas en componentes y frameworks para la inyección y representación de servicios. A su vez, se observa que el camino de su instrumentación es complejo debido a que se deben generar (automáticamente) porciones de código y creación de nuevos archivos código Java, a partir de los originales de la plataforma Sakai, que deberán ser compilados y puestos en servicio para su operatividad. Cabe destacar que los procesos de compilación y puesta en servicio de Sakai es efectuada una sola vez, cualquier tipos de cambios y configuración serán impuestas a través los contratos sensibles al contexto.

La implementación de la coordinación de contratos en Sakai requiere del uso de una herramienta específica para el dominio de aplicación (en este caso aplicaciones e-learning) que modifica los archivos de código fuente Java que implementan los servicios bases perteneciente al framework núcleo (similares a los representados en esta sección), con el fin de incorporar la infraestructura para trabajar con contratos de forma más o menos automática.

Para este propósito, se diseñó e implementó un prototipo experimental de una herramienta llamada SwC (“Sakai with Contract”) y un entorno de desarrollo que permite instrumentar una metodología (? 60) para la inclusión de contratos en

Sakai bajo la perspectiva de un DHD. El desarrollo de la herramienta SwC está basada sobre el proyecto CED (Coordination Development Environment)<sup>d</sup> que implementa la coordinación de contrato a través de un lenguaje llamado Oblog (72). En este artículo no se describen detalles técnicos de cómo la herramienta lleva a cabo la modificación del código Java dentro de los distintos componentes de la arquitectura Sakai y cuáles fueron las modificaciones puntuales efectuadas a partir de CED.

## 6.5. Caso de estudio para la implementación de contratos

En esta sección se presentará un caso de uso para ejemplificar las diferentes etapas que se deben cumplimentar para lograr la inclusión de contratos en Sakai bajo la perspectiva de los DHD, retomando la idea de incluir un contrato en los servicios de edición de la herramienta Foro de Sakai.

Si bien consideramos importantes las observaciones realizadas sobre el por qué de la necesidad de utilizar una metodología concreta para el diseño del contrato referidas al lugar que ocupa dentro del flujo de ejecución de un Pe-Irn, qué tipo de requerimientos satisface, cómo fueron relevados dichos requerimientos y una especificación (en lo posible semi-formal) para su implementación, un desarrollo más exhaustivo consta en otra publicación. Dicha metodología -llamada UWATc- fue desarrollada en (60) para este mismo caso de estudio, y en (?) se amplían los detalles tecnológicos involucrados en su diseño. En esta instancia UWATc brindará información sobre: cuáles son los servicios, en qué clase (archivo Java) se encuentran representados (teniendo en cuenta el flujo de ejecución) y cómo es configurado el contrato.

Nos centramos ahora en la última etapa del ciclo de vida de la configuración de un espacio e-learning Web perteneciente al tiempo de compilación. En efecto, en esta instancia el ingeniero de software cuenta con la información necesaria para localizar los archivos de código fuente de la aplicación Sakai, los archivos de configuración de los frameworks adicionales y demás configuraciones propias de la herramienta para la inyección de los contratos.

Siguiendo con el caso de estudio de las etapas anteriores, a continuación se muestra una porción de código Java correspondiente al código fuente original de Sakai, en donde se propone un método llamado **editMensaje** correspondiente a la implementación del servicio edición de la herramienta Foro. El siguiente fragmento de código fuente Java implementa una herencia de la clase *BaseDiscussionService* donde se encuentran las interfaces de los servicios de edición correspondiente al núcleo del framework Sakai.

---

<sup>d</sup>CED: es el primer prototipo de una herramienta que implementa el uso de la coordinación de contrato en aplicaciones Java. La herramienta pertenece a ATX Software ([www.atxsoftware.com](http://www.atxsoftware.com)); fue desarrollada en Java y es de código abierto.

```
import org.sakaiproject.discussion.api.DiscussionMessage;  
public class DiscussionService extends BaseDiscussionService{  
public MessageEdit editMessage(MessageChannel channel, String id){  
return (MessageEdit) super.editResource(channel, id);}  
}
```

Luego, a través de la herramienta SwC se crean archivos XML (similar a la herramienta CED) para especificar las reglas del contrato que involucrarán servicios implicados en el método *editMensaje* (Paso1). A continuación, se ejecuta la generación automática de código para crear dos tipos de archivos Java diferentes (Paso2). El primero (archivo 1) implementa las funcionalidades que permitan las conexión con el Proxy (ej., *MétodoHerramienta\_Proxy* para nuestro caso de estudio); el segundo (archivo 2) implementa el conector *ConectorEditService* representado en la figura 6.3.

A continuación se muestran fragmentos de código que permiten una mejor ilustración de las características adquiridas a partir de las modificaciones del código fuente de la aplicación Sakai a través de la herramienta.

### Fragmentos del primer archivo - (archivo 1)

1. Se importan los paquetes correspondiente al framework de coordinación de contratos (figura 6.3) y el framework context-aware (figura 6.1)

```
package org.sakaiproject.discussion.impl; import java.util.*;  
import cde.runtime.*; import obab.ca.*; // Framework context  
public abstract class DiscussionService extends  
BaseMessageService implements  
DiscussionService,ContextObserver,EntityTransferrer,ForoInterface
```

2. Métodos agregados por la herramienta para identificar las clases que van a ser interceptadas por el contrato.

```
protected CrdIPProxy _proxy;  
private static Class _classId= Sakai.Discussion.class;  
public static Class GetClassId() {return _classId;}  
public CrdIPProxy GetProxy() { return _proxy; }  
public void SetProxy(Object p){if(p instanceof CrdIPProxy && p instanceof  
DiscussionInterface) _proxy = (CrdIPProxy)p; }  
public void SetProxy(CrdIPProxy p) { _proxy = p; }  
AccountInterface GetProxy_Account(){if ( _proxy == null ) return null;  
return (DiscussionInterface) _proxy.GetProxy(_classId);}
```

3. Método que implementa la llamada del objeto cliente al Proxy

```
public long _getNumber(){new ComponentOperationEvent(this , "getNumber")_
    .fireEvent(); return number;}
public messageEdit editMessage(MessageChannel channel,String id){
    new ComponentOperationEvent(this,"Edit").fireEvent();
    return (MessageEdit) super.editResource(channel, id);}
```

### Fragmentos del segundo archivo - (archivo 2)

1. Porción de código donde se importan las componentes de los frameworks, se heredan las clases abstracta del los conectores y proxys. La clase del objeto real *MétodoService\_Componente*, según la figura 6.3, se representa a través del atributo subject.

```
package org.sakaiproject; import java.util.*;
import cde.runtime.*; import obab.ca.*;
public abstract class IDiscussionPartner extends
    CrdContractPartner implements CrdIProxy, DiscussionInterface {
    protected Discussion subject;
```

2. Definición de los métodos abstractos para la conexión (*ConnectorMétodoService*, representada en la figura 6.3), del contrato con el servicio.

```
public void SetProxy(Object p) {subject.SetProxy(p);}
protected Object GetSubject_Object() {return subject;}
public void ResetProxy() { subject.SetProxy(null);}
```

3. Métodos que permiten el acceso a los métodos que definen los servicios (ej., métodos de la clase *MétodoService\_Componente*)

```
protected Discussion GetSubjectDiscussion(){return (Discussion) subject;}
protected IDiscussionPartner GetNextPartner_Discussion(){
    return (IDiscussionPartner)GetNextPartner(Discussion.GetClassId());}
```

4. Implementación por defecto de métodos definidos en la interfaces de los servicios. A través del métodos *GetSubjectDiscussion()*, detallado en el punto anterior, se accede a los métodos creados por la herramienta en el primer archivo.

```
public void messageEdit (double amount, Customer c)_
    throws DiscussionException { IDiscussionPartner
    next = GetNextPartner_Discussion()
    if (next != null) next.editMessage(amount,c);
    else GetSubjectDiscussion()._editMessage(amount,c);}
```

5. Implementaciones de los condicionales de la reglas de los contratos que se encuentran en los archivos XML para configurarlos.

```
public CrdPartnerRules messageEdit_rules(string texto, Student c) throws  
DiscussionException, CrdExFailure { return new CrdPartnerRules (this);}
```

La generación de código automático a través de la herramienta presupone tener ciertos niveles de conocimientos sobre: el lenguaje Java, aspectos de la implementación del framework base Sakai y los frameworks que los componen; a igual que experiencias en la compilación a través de Maven <sup>e</sup> de Sakai y las clases modificadas-agregadas para el uso de contratos.

## **6.6. Conclusión**

Fundamentados en el marco conceptual de los Dispositivos Hipermediales Dinámicos para educación e investigación y en las actuales limitaciones observadas en las plataformas e-learning sobre el grado de flexibilidad adaptativa en el sentido de (80) y (73)), que pueda ser inducida por usuarios expertos del dominio (ej., docentes) para el desarrollo de estrategias didácticas efectivas para el logro de objetivos pedagógicos o investigativos planteados y, constando que dicha adaptabilidad no pueda ser enteramente resulta por sistemas adaptativos inteligentes (ej., agentes, hipermedia adaptativa, sistemas expertos, etc.), es que en este trabajo propusimos una implementación de la teoría de coordinación de contrato en un framework específico sobre sistemas colaborativos Web e-learning (correspondiente al proyecto Sakai) brindando un modelo evolutivo conceptual y tecnológico. Partimos entonces, de las implementaciones de servicios y herramientas, consideramos el agregado de componentes para la representación de cierta información de contexto con aspectos context-aware para finalizar con la incorporación de contratos para la coordinación de las interconexiones de los servicios bases del framework original e-learning. De esta manera, el contrato es una nueva primitiva prevista como una extensión de la noción de contratos de B. Meyer (70) a la que se le adjudica el rol de coordinación de las interacciones de los objetos.

El fin último de esta propuesta en desarrollo, se inscribe en brindar respuestas tecnológicas efectivas a la necesidad de promover procesos educativos e investigativos de interacción responsable entre los sujetos intervinientes en la red sociotécnica del DHD, implementando la solución en los servicios cuya prestaciones dependen -fuertemente- de reglas con estructuras volátiles cuyos condicionales son

---

<sup>e</sup>Proyecto maven: <http://maven.apache.org/ref/2.0.4/maven-project/>

## 6.6. Conclusión

---

influidos por el contexto (en el sentido de "Identificación del contexto" desarrollado en el capítulo 5 (57) y manteniendo las lineamientos sobre contexto fundados por P. Dourish (79)).

En este trabajo se propuso una implementación de la teoría de coordinación de contrato en un framework específico sobre sistemas colaborativos Web e-learning (correspondiente al proyecto Sakai). Se brindó un modelo evolutivo conceptual y tecnológico, comenzando por las implementaciones de servicios y herramientas; luego con el agregado de componentes para la representación de cierta información de contexto con aspectos context-aware; finalizando con la incorporación de contratos para la coordinación de las interconexiones de los servicios bases del framework original e-learning. De esta manera, el contrato es una nueva primitiva prevista como una extensión de la noción de contratos de B. Meyer (?) a la que se le adjudica el rol de coordinación de las interacciones de los objetos.

Basados en nuestra visión de que las plataformas e-learning actuales carecen de ciertos grados de flexibilidad adaptativa (en el sentido de (80) y (73)), que pueda ser inducida por usuarios expertos del dominio (ej., docentes, expertos en educación), para construir procesos educativos con cierto grado de eficiencia. Donde, dicha adaptabilidad no pueda ser enteramente resulta con sistemas adaptativos inteligentes (ej., agentes, hipermedia adaptativa, sistemas expertos, etc.). Este requerimiento puntual hace referencia a las necesidades de promover el tipo de solución que en este trabajo se propone. Implementando la solución en los servicios cuya prestaciones dependen -fuertemente- de reglas con estructuras volátiles cuyos condicionales son influidos por el contexto (en el sentido de "Identificación del contexto" desarrollado en el capítulo 5 (57) y manteniendo las lineamientos sobre contexto fundados por P. Dourish (79)). De esta manera se propone un mecanismo para el diseño e implementación en tiempo de ejecución adaptada a un tipo de aplicación e-learning Web.

---

## Implementaciones del framework DHD: Casos de usos

---

### **7.1. Usos de los contratos sensibles al contexto: Evidencias en el uso de entornos E-learning**

A medida que el avance en la investigación y desarrollo de plataformas e-learning brinden mejoras e innovaciones de herramientas (videoconferencias, portfolios, wikis, workshops, etc.) y sus respectivos servicios, crece la cantidad de posibles configuraciones de los espacios e-learning. Abarcando diferentes tipos de requerimientos pertenecientes a las etapas de diseño, otros durante el desarrollo y otros en tiempo de ejecución. A partir de estos se definen los procesos e-learning (Pe-Irn). Al igual que los procesos de negocios en una Aplicación Web convencional (ej., [www.ebay.com](http://www.ebay.com)), los Pe-Irn están compuesto por transacciones Web (? ). En este contexto, una transacción (o transacción e-learning) es definida como una secuencias de actividades que el usuario ejecuta a través de una Aplicación E-learning con el propósito de efectuar una tarea o concretar un objetivo. El conjunto de actividades, sus propiedades y las reglas que gobiernan sus ejecuciones dependen del Pe-Irn que la Aplicación debe brindar.

Las características tecnológica de un Aplicación Web e-learning (AWe-Irn) son idénticas alas Aplicaciones E-learning convencionales (por ejemplo, las utilizadas



## 7.1. Usos de los contratos sensibles al contexto: Evidencias en el uso de entornos E-learning

---

en el proyecto de e-learning Sakai <sup>a)</sup>

que proveen: navegación entre páginas a través de links, ejecución de Transacciones e-learning por medio de los servicios de las herramientas y las operaciones de un Pe-Irn. La principal diferencia tecnológica entre un AWe-Irn y una Aplicación e-learning convencional está en la incorporación de la teoría de coordinación de contratos (68; 58) en la implementación de algunos servicios que las herramientas brindan a los usuarios (57).

En base a la incorporación de los contratos en la implementación y diseño de Transacciones e-learning, aumenta las posibles configuraciones de los Pe-Irn. Comienzan a aparecer nuevas propiedades que tienen que ver con el campo de la Ingeniería de Software. Y, a su vez, fuertemente relacionados al trabajo multidisciplinario entre los actores de un proyecto de construcción de una AWe-Irn (ej. el proyecto Obra Abierta <sup>b)</sup>). Para este fin, es imprescindible contar con un modelo de diseño que ayude en el ciclo de vida del desarrollo y configuración de una Aplicación e-learning. Integrando las tareas de los expertos en educación, diseñadores y desarrolladores.

Desatender o resolver incorrectamente la documentación para el diseño de procesos tipo Pe-Irn pueden causar numerosos problemas reflejados en el proceso de configuración de un AWe-Irn. Estos problemas pueden ser: (1) Dificultades en la comunicación y entendimiento entre los clientes y diseñadores expertos en educación (primero), y entre los diseñadores y los desarrolladores (después), en el proceso de implementación de un AWe-Irn. (2) Determinación de las relaciones donde se justifique la inclusión de contratos. (3) Dificultad para visualizar la trazabilidad entre los procesos e-learning y las implementaciones de las Transacciones.

Este trabajo presenta un diseño compresivo para el modelado de los procesos de educación e-learning (Pe-Irn) en un Aplicación Web E-learning con la inclusión de los contrato con propiedades context-aware (57). El modelo está basado en UWAT+ (Distante, 2004), una versión extendida y adaptada de "ÜWA Transaction Design Model" para el diseño de transacciones en aplicaciones Web. La principales contribuciones de este trabajo son: - El acercamiento de un modelo útil para la representación de transacciones e-learning en una AWe-Irn; permitiendo una mejor distinción de la ubicación (entre servicio-usuario(s) y servicio-servicio(s)) de la componente contrato dentro del flujo de ejecución. - Ejemplificar, mediante un

---

<sup>a</sup>Sakai: Entorno colaborativo y de aprendizaje para enseñar. Es de código abierto y está resuelto con tecnología Java. Url: [www.sakaiprojet.org](http://www.sakaiprojet.org)

<sup>b</sup>Obra Abierta: Proyecto de ID (CONICET), que se centra en el desarrollo e implementación de dispositivos hipermediales context-aware dinámico para investigar y aprender en contextos físicos-virtuales de educación superior. Directora: Patricia San Martín

caso de uso concreto, el uso del modelo UWATc+.

El resto de este trabajo se encuentra organizado de la siguiente manera: Tras esta introducción se presenta una referencia conceptual del los contratos, utilizados en el proyecto Obra Abierta (Sección 6.2). Luego se continúa con definiciones, formas de documentación y representación de Transacciones e-learning y procesos. (sección ??). Siguiendo, se presenta un modelo conceptual y de diseño sobre la adaptación propuesta para el modelado de Pe-Irn (sección ??). Mediante un caso de uso concreto, se describe el uso de UWATc+ para cada una de sus etapas. Finalizando con una breve conclusión.

### 7.1.1. Contratos context-Aware para transacciones en los DHD

En términos generales, la coordinación de contratos es una conexión establecida entre un grupo de objetos (en nuestras consideraciones, un objeto cliente y un determinado servicio). Cuando un objeto cliente (ej. un usuario alumno) efectúa una llamada a un objeto suministro (ej. un servicio de la herramienta Foro), el contrato "intercepta" la llamada y establece una nueva relación teniendo en cuenta el contexto del objeto cliente, el del objeto servidor, e información relevante adquirida y representada como contexto del entorno. Como condición necesaria, la implementación de los contratos no debe alterar el diseño y funcionalidad en la implementación de los objetos.

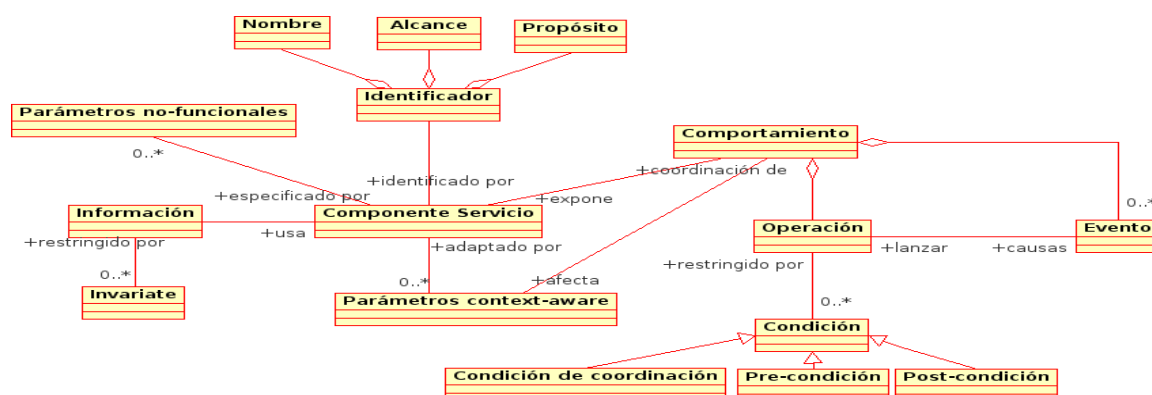
#### Elementos de la componente contrato

El contrato puede ser configurado por medio de diferentes mecanismos, desde el lenguaje cotidiano hasta un lenguaje de especificación formal y un lenguaje basado en XML para los casos que sean necesarias especificaciones que puedan ser procesadas por máquinas.

El tipo de tecnología y forma de implementación de los contratos es transparente para los objetos que consumen los servicios en donde se encuentran involucrados. La configuración de un elemento contrato que forma parte de las componentes de un servicio, representa la información necesaria del mismo para ser utilizado por el invocador, sin necesidad que el dicho objeto tenga detalles de la ejecución. El contrato representa una enriquecida y efectiva interface de construcción que contiene toda la información sobre las componentes de los servicios, y deberá tener referencias sobre algún tipo de información de contexto para su utilización.

## 7.1. Usos de los contratos sensibles al contexto: Evidencias en el uso de entornos E-learning

En la figura 6.1 se puede observar los elementos conceptuales básicos de esta componente a través de una serie de elementos que lo relacionan.



Para una mejor comprensión de las componentes del modelo se explica su caracterización y funciones particulares:

*Identificador* – Una componente servicio es identificada para un determinado contexto por un único nombre en el espacio de nombre.

*Comportamiento* - De acuerdo con los roles asignados en un determinado contexto, una componente servicio expone comportamientos correspondientes a la provisión y pedido de operaciones, y/o publicaciones y recepción desde/hacia cada contexto. Las operaciones pueden ser definidas en dos tipos – operaciones que ejecutan cálculos o transformaciones (tipo “update”) y operaciones que proveen algún tipo de información sobre consultas (tipo “query”). Estas, se encuentran enteramente especificadas en base a un contrato, con el uso de pre-condición, post-condición y condicionales para lograr la coordinación entre contratos. En las condiciones de coordinación se especifican cómo requerir y proveer operaciones. Para lograr una comunicación precisa con una componente servicio, no sólo se tiene en cuenta qué operación fue provista o requerida y cómo el ejecutor ha lanzado el evento apropiado, sino también, cómo todas esas actividades están mutuamente relacionadas para ser aprovechadas por el objeto cliente. Un evento del contexto que lanza una operación dada, puede ser parte de un conjunto de pre-condición, mientras que un evento emitido a través de una operación exitosa puede ser parte de una pos-condición.

*Tipos de Información* – Una componente de servicio debe manejar, usar, crear o tener cierta información de recursos con el propósito de proveer servicios adecuadamente. Este elemento del contrato define el tipo de información relevante para las componentes asociadas al contrato, así como también restricciones y reglas sobre instancias de esos tipos. Esto representa un modelo de información lógica de una componente de servicio. Formalmente, esta información de tipos puede ser

considerada como definiciones de tipos de los parámetros de las operaciones, o tipos relacionados a ellos.

*Configuración de Parámetros Context-Aware* – Una componente servicio depende del contexto de su actual entorno. Para utilizarse en diferentes contextos logrando la adaptación a eventuales cambios debe tener definido un conjunto de parámetros de configuración. Ejemplos de estos parámetros pueden ser: Contexto-del-Usuario (CU), locación en tiempo y espacio de los servicios consumidos y suministrados. Estos parámetros pueden ser enviados dentro de las invocaciones de las operaciones de los servicios o por medio de otros caminos, mediante componentes de servicios que pueden adaptar su comportamiento ante el cambio de contexto en una determinada situación.

*Parámetros no funcionales* – Una componente servicio puede definir un conjunto de los llamados parámetros no funcionales que caracterizan a la “calidad” de sus prestaciones dentro de un determinado contexto. Estos parámetros, son elementos para los consumidores de los servicios que permiten optar por el uso de un determinado servicio, o buscar otro con el mismo o similar contrato. Como ejemplo de parámetros no funcionales podemos mencionar: Performance, Fiabilidad, Tolerancia a Fallos, Costos, Prioridad y Seguridad.

### 7.1.2. Documentación de los DHD procesos

Las transacciones e-learning en un AWe-Irn están definidas como secuencias de actividades asociadas con un flujo de ejecución que permite al usuario desempeñar una determinada tarea y/o alcanzar una meta a través de la Aplicación. Entonces, un proceso e-learning (Pe-Irn) puede ser interpretada como una especificación del concepto de “workflow.”<sup>en</sup> una Aplicación e-learning Web, con las condiciones (restricciones) que implica su concreción. En una Transacción e-learning Web, una *Actividad* está conformada por un conjunto de operaciones simples o complejas sobre datos y contenidos de la Aplicación. Como ejemplo de Transacciones e-learning se pueden mencionar un proceso en el cual un usuario (alumno) participa en un espacio de edición colaborativa (este caso será analizado posteriormente como caso de uso en las secciones posteriores)

Entonces, las transacciones en un AWe-Irn son el camino para la representación de los Pe-Irn, y proveer a los usuarios de servicios accederlos por medio de las herramientas que los contienen (wiki, foro, vídeo conferencia, taller, blog, etc.) La ejecución de Transacciones de un AWe-Irn supone tanto la navegación a través de las herramientas, por medios de los links de las componentes hipermediales, como

## 7.1. Usos de los contratos sensibles al contexto: Evidencias en el uso de entornos E-learning

---

el uso de sus servicios. Un ejemplo de servicio puede ser en una vídeo conferencia la posibilidad de que un docente edite en una pizarra compartida (con sus alumnos y colaboradores) una determinada ponencia.

El diseño de las transacciones abarca varios niveles de abstracción, distintos formalismos pueden ser usados para su representación y documentación. Al menos tres niveles de abstracción pueden ser representados: (1) nivel conceptual, (2) nivel lógico, y (3) nivel de implementación. El diseño conceptual permite una representación del sistema (las transacciones y su alcances) tal cual son "percibidas" por el usuario y despejando las cuestiones de implementación. El diseño de la implementación se encuentra focalizado a proveer a los diseñadores de los AWe-Irn con todas las especificaciones necesarias para la configuración y realización de sus componentes. El diseño lógico es un nivel intermedio del diseño de abstracción, utilizado para trasladar las especificaciones centrales del usuario desde el diseño conceptual hacia términos de especificaciones más cercana a las implementaciones.

Al igual que lo que ocurre con todos los artefactos de software, el diseño de una Transacción e-learning para un AWe-Irn se puede tornar muy complejo. Cuanto más complejo sea el diseño, las confusiones entre los diseñadores (expertos en educación y analistas informáticos) y los implementadores (programadores y encargados de la Aplicación) crecerán. Para comunicar efectivamente la idea del diseñador es necesario una apropiada documentación. Si bien la documentación textual ha sido muy utilizada para describir detalles de implementaciones de bajo nivel, teniendo en cuenta que el diseño de las transacciones e-learning se describen en un nivel conceptual, es más adecuado una representación gráfica.

Existen diferentes aportes directamente relacionado a modelos "visuales" en forma de documentación gráfica (62; 63; 64). En este contexto los modelos visuales son representaciones de sistemas de software que soportan múltiples perspectivas. Para el caso del diseño de las transacciones e-learning, una vista puede ser representada por una serie de diagramas pertenecientes a UML (Unified Modeling Language) ( ? )

Los antecedentes relevantes que se relacionan con lo que entendemos por diseño de Pe-Irn, fueron estudiados de los aportes en el campo de los métodos de diseños para Aplicaciones Web experimentados en los últimos años. Concretamente se pueden mencionar ADM (Atzeni y Parente, 2001), OO-H (Koch et al., 2003), OOHDM (Schmid - Rossi, 2004) y UWAT+ (Distante, 2005).

UWAT+ es un meta-modelo para la descripción de los distintos aspectos de

Transacciones Web de manera holística. Es una extensión del Modelo de Diseño de Transacciones que forma parte del framework UWA (Ubiquitous Web Applications) (65). Inspirado en este modelo y extendiéndolo para la contención de los contratos, se describe una adaptación para el diseño de Transacciones e-learning en un AWe-Irn.

### 7.1.3. UWATc+: una adaptación de UWAT+ para el modelado de DHD procesos

Si bien los métodos de diseño de Transacciones de UWAT+ pueden ser utilizados para la representación de Transacciones e-learning, es necesario efectuar adecuaciones que tengan en cuenta la inclusión a los contratos (según sección 6.2). Tal cual fue mencionado en la sección 6.1, desde la perspectiva de los diseñadores, los contratos deben ser visto como una pieza de software para la instrumentación de los servicios de las herramientas. En consecuencia, es necesario tener un modelo que permita una mejor representación de los contratos, la visualización de su inserción en los servicios y las relaciones que en ellos representan (relaciones entre objetos que implementan servicios, usuarios y herramientas en la Aplicación).

La figura ?? se muestra un diagrama de clase UML que representa a los conceptos, las relaciones entre conceptos y los modelos para la representación de Transacciones. Los esquemas en color blanco pertenecen al modelo original UWAT+. El rectángulo y los esquemas grises describen los objetos, modelos y relaciones que conforman el nuevo modelo denominado UWATc+.

Como se describe en el diagrama, una Transacción Web es un objeto complejo (conceptual) compuestos por dos tipos de objetos principales pertenecientes al modelo original de UWAT+ y un tercer objeto agregado para la representación de los contratos pertenecientes a las Transacciones e-learning. En el primer grupo se encuentra *Actividad* para la distinción de las actividades de los usuarios y del sistema. El objeto *FlujodeEjecución* representa el orden lógico y temporal para la ejecución de las actividades comprendidas en las Transacciones. A su vez, una Transacción Web puede ser descripta por el *ModeloOrganizacional* (desde el punto de vista estático) y el *ModelodeEjecución* para la definición de las reglas de ejecución de la componente actividad (desde el punto de vista dinámico).

Cuando una Transacción Web contiene un contrato (definida como transacción e-learning) debe ser incluida una nueva componente para el diseño (representada en la figura como una relación de agregación en el *ModelodeEjecución*), conjuntamente con un nuevo modelo de diseño, *ModelodePe – Irn*, que permitirá

## 7.1. Usos de los contratos sensibles al contexto: Evidencias en el uso de entornos E-learning

---

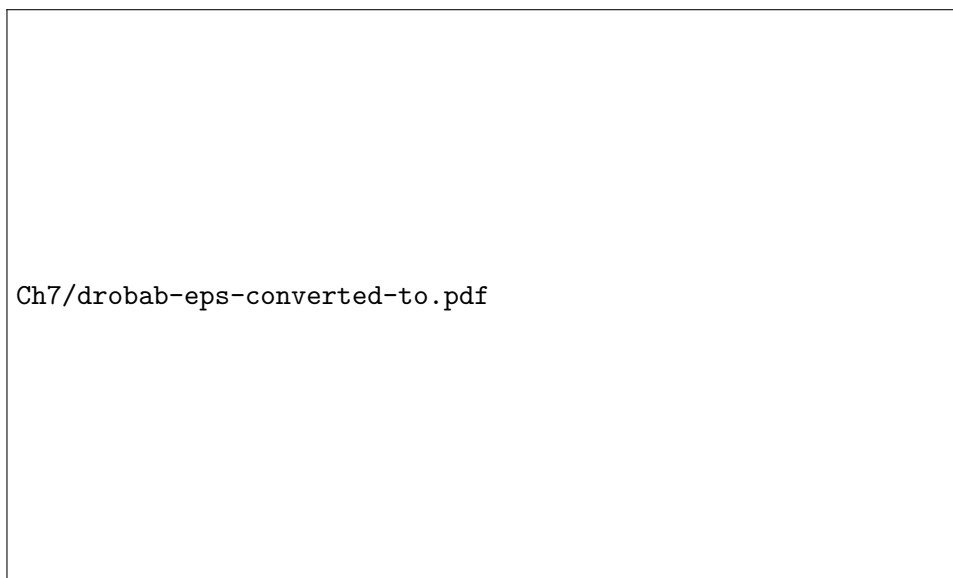


Figura 7.1: *UWATc+ modelo conceptual y de diseño*

la representación del contrato (caracterizada como relación de asociación con el objeto *Contrato*).

De esta manera quedan conformados los elementos que componen el modelo UWATc+ (rectángulo gris) y sus relaciones con el modelo original UWAT+. A continuación se describe, en detalle, el modelo usado por UWATc+ para el diseño de las transacciones que utilizan contratos (transacciones e-learning).

### Requerimientos para el diseño de DHD Transacciones

En esta sección se describen la caracterización de dos tipos representativos de requerimiento que motivaron la creación de este nuevo modelo de diseño (definidos en la sección 6.1). En base a experiencias recogidas por el grupo del proyecto Obra Abierta en el diseño y configuración de Aplicaciones e-learning, se presentarán dos tipos de requerimientos que deben ser cubiertos por el modelo de diseño. En primer lugar se enuncian cuestiones técnicas de diseño (desde el punto de vista de la Ingeniería de Software), seguido de un comentario sobre el tipo de Transacciones que el diseñador puede especificar a través del modelo.

- Especificar como son afectadas la ejecución de las actividades por los objetos contratos.

La relación de una actividad con un contrato se produce cuando existen objetos interrelacionados por medio de un contrato (68). De esta manera, el

diseñador debe poder documentar la información de los objetos involucrados, sus métodos y parámetros. El contrato representa un tipo diferente de relación a la original entre los objetos, con la propiedad de reconfiguración en tiempo de ejecución. Característica que permiten una mejor adaptación a los requerimientos que resuelven las Transacciones e-learning.

- Definir cuál y cómo la información de los objetos contratos (information object" ( ? )) es afectada por la ejecución de las Actividades.

Una actividad funcional consiste en la ejecución de uno a más operaciones elementales (inserción, borrado, modificación, etc.) sobre los datos de la Aplicación y la información de los objetos contratos envuelta en la Actividad. El modelo de diseño de Transacciones e-learning debe permitir al diseñador definir cuales operaciones del contrato son fundamentales para cada actividad, modelando el camino en que cada actividad elemental afecta la información de los objetos involucrados (modificando sus instancias por medios de sus ejecuciones).

### 7.1.4. Diseño de procesos e-learning con UWATc+

En esta sección se describen los resultados de la Aplicación de UWATc+ para el espacio dedicado al libro "Hacia un dispositivo hipermedial context-aware Dinámico. Educación e investigación para el campo audiovisual interactivo" (San Martín P. 2007, et. al) (57). Este modelo fue aplicado parcialmente para el diseño de los requerimientos fundamentales y el modelado del comportamiento funcional enmarcado bajo la perspectiva de proyecto Obra Abierta - mencionado anteriormente. Se presentará un caso de uso para ejemplificar el diseño de un Pe-Irn en la Aplicación e-learning para Obra Abierta. El diseño describe parte del proceso en que un usuario hace uso de los servicios de edición de la herramienta Foro a través de un contrato.

El diagrama de la figura ?? muestra una adaptación del diseño de procesos de UWA (65) que se utiliza en UWATc+ para el diseño de transacciones con contratos. Para ilustrar el proceso de diseño se utiliza la notación IDEF0 (IDEF-0, 1993). En comparación con la metodología original de UWAT+, se agregaron las fases del diseño de contrato y fue modificado el modelo de diseño de las transacciones. Además, fueron excluidas las fases de "diseño de la información" "diseño de publicación". Las fases del proceso de diseño quedan conformadas de la siguiente manera: (1) Determinación de Requerimientos; (2) Diseño de transacciones con contrato; (3) Diseño de contrato.



## 7.1. Usos de los contratos sensibles al contexto: Evidencias en el uso de entornos E-learning

Ch7/procesodisenoNuevo-eps-converted-to.pdf

Figura 7.2: El proceso de diseñar Pe-Irn en un AWe-Irn con UWATc+

### Determinación de Requerimientos

La fase de Determinación de Requerimientos toma como entrada las especificaciones del proyecto y produce, por medio de un mecanismo de refinamiento, las siguientes salidas: Cada actor con su objetivos relativo. Los requerimientos para la construcción y configuración de un AWe-Irn.

El modelo utilizado es orientado a objetivos: cada actor se identifica con al menos un objetivo, i.e, una abstracción de los objetivos que a través de la aplicación se debe alcanzar; cada objetivo es refinado en otros sub-objetivos, hasta poder definir el requerimiento en un bajo nivel suficiente para poder ser implementado. Este fase es similar a la descrita en (UWA Consortium, 2001) (65).

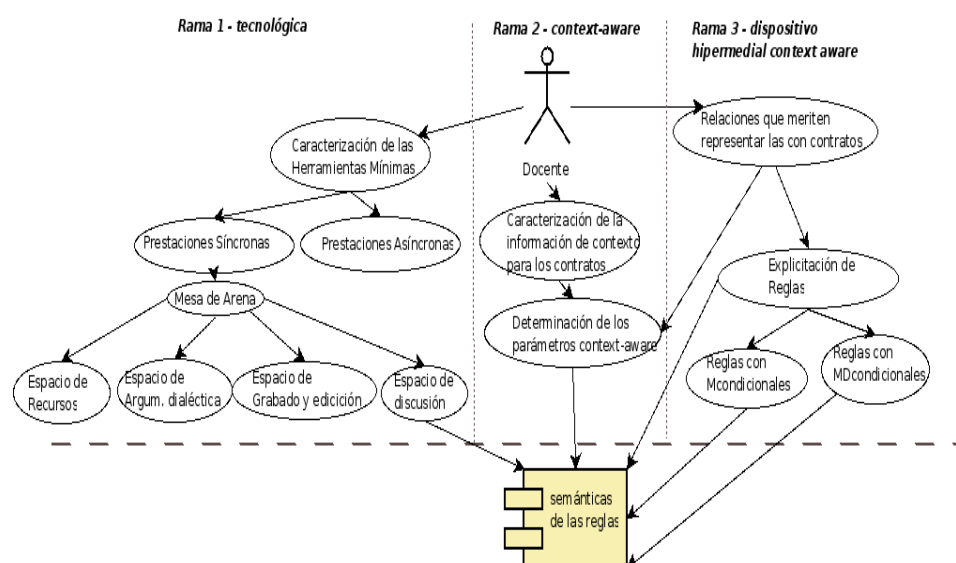


Figura 7.3: Objetivos de al alto nivel para un Pe-Irn

Para este caso de uso, atendiendo a los lineamientos en (57), se describe parte del modelo original donde se caracterizan los objetivos involucrado con un actor del sistema. A su vez, a medida que se van derivando los sub-objetivos comienzan a establecerse requerimientos concernientes a la teoría de coordinación de contratos context-aware (58; 68). En la figura ?? dicha situación ocurre en la derivación de las tres ramas de objetivos y sub-objetivos, influyendo directamente en la composición de la componente contrato. En este caso, tomando desde la **rama 1** un servicio de una herramienta de un espacio de discusión (herramienta Foro); de la **rama 2** se desprenden la información necesaria para poder articular dicho servicio teniendo en cuenta la información de contexto; la **rama 3** aporta el consenso de los expertos (del dominio e-learning) para la inclusión de los contratos en aquellas relaciones que mantendrán las propiedades de la Aplicación e-learning (58). A través de este modelo, se logra un primer acercamiento sobre como se relaciona un contrato con: los requerimientos, el tipo de objetivos para cada requerimiento y los actores del sistema.

### Diseño de DHD-Transacciones

El diseño de transacciones e-learning retoma la misma idea y modelo propuesto por UWAT+ para el diseño de transacciones (? ). Partiendo de los resultados de la fase de *Determinación de Requerimientos*, fundamentalmente de la caracterización de los contratos, pueden ser seleccionados una series de transacciones, i.e., objetivos que requieran la ejecución de una o más Actividades para su cumplimiento. Para cada uno de los objetivos que incluya contrato deben ser diseñadas Transacciones e-learning (de igual forma que con las transacciones en UWAT+), los cuales en principio deben ser establecidos desde un punto de vista estático (en este trabajo no abordaremos tal consideración) y luego, desde un punto de vista dinámico por medio de un modelo de ejecución. En la figura ?? se muestra una porción del modelo de ejecución de un transacción e-learning cuyo contrato asociado fue caracterizado en la fase de *Determinación de Requerimientos* (figura ??, sección ??). En el diseño se describe el flujo de ejecución entre las Actividades de la transacción. El modelo de ejecución es una adaptación del diagrama de actividades de UML (? ) en el que las Actividades y sub-Actividades están representadas por estados (óvalos), y el flujo de ejecución entre ellos se representa por medio de transiciones (arcos). Los óvalos con el símbolo (\*) - un asterisco entre paréntesis - refiere a una Actividad que representa a un conjunto de Actividades compuestas, y cuyo modelo de ejecución debe ser representado con otro diagrama. Un óvalo simple representa una Actividad Elemental. Un óvalo color gris indica una Actividad compuesta de las sub-actividad que se encuentra dentro. Una sub-Actividad representada con un óvalo color gris indica que es dependiente de la

## 7.1. Usos de los contratos sensibles al contexto: Evidencias en el uso de entornos E-learning

---

Actividad que la contiene, esto quiere decir que su ejecución estará acompañada por otra sub-Actividad y no puede ser incluida en otra composición. Los arcos de líneas continuas indican flujos de ejecución obligatoria (transacciones hacia Actividades requeridas), mientras que los actos con líneas de puntos representan flujo de ejecuciones opcionales (transacciones hacia Actividades opcionales).

Cada relación posible entre actividades es representada por medio de una arco entre ellas. A cada arco se le asocia un texto que indica bajo que condiciones se produce la transacción, o el resultado de la ejecución de la Actividad de origen. Para describir como colaboran los usuarios de la aplicación en la ejecución de la Actividades puede ser anexado un diagrama UML Swimlanes.

Cuando una Actividad ejecuta servicios implementados por contratos, entonces, se establece un arco saliente hacia un contrato. El contrato tiene un nombre, y entre paréntesis se indican cuales son los objetos participantes (en el caso de tener ese tipo de información). Para representarlo visualmente se utiliza el estereotipo del elemento componente de UML. Las Actividades que influyan en la modificación de los contratos en tiempo de ejecución se conectan a través de un arco de línea de puntos, igual a los utilizado en la representación de los flujos opcionales. Los detalles de implementación del contratos se detallan en un diagrama aparte, perteneciente a la fase descrita en la siguiente sección.



Figura 7.4: *Modelo de ejecución de Procesos con Transacciones e-learning*

Por ejemplo, una de las opciones de la herramienta Foro de Obra Abierta (<http://200,80,157,171:8080/portal>) es la visualización de las intervenciones de los usuarios en el Foro. Un usuario docente puede seleccionar la opción "Foro" de la página principal de la Aplicación, luego seleccionar el tipo de "vista" (mediante un `comboBox`) para ingresar en modo "browser" donde se muestran las intervenciones de los usuarios por temas. Una vez seleccionado el tema (por medio de la Actividad "Seleccionar Tema", figura ??), es posible ingresar al espacio de las intervenciones de los usuarios por medio de los roles de docente o alumno. Los docentes y alumnos tienen diferentes tipos de "vistas", permisos y servicios asociados (representadas en las actividades: "Visualización de Intervenciones Docentes" y "Visualización de Intervenciones Alumnos"). A través del Browser (Actividad "Foro Browser") se recorre todo el contenido del espacio y al mismo tiempo puede ser seleccionado otro tema para visualizar.

En cambio, si la opción seleccionada es añadir o responder temas, se ingresa a una página Web configurada para editar texto (por medio de la Actividad "Adquirir servicios de escritura"). Algunos de los servicios de edición y configuración de opciones son implementadas a través del contrato "Edición" (representado por la figura de la componente UML, con el contorno color gris). El flujo de ejecución, luego de la intervención de los contratos, dependerán de las reglas de coordinación y se representan con los arcos salientes similares a los usados para representar las relaciones entre estados.

### Diagrama de un Contrato

Existen diferentes formas de representación de los contratos definidos en la sección 6.2, la herramienta CED (Coordination Development Environment) <sup>c</sup> los implementa a través de un lenguaje llamado Oblog (72). En (98) se muestra como a través de CommUnity se definen primitivas de modelado y técnicas de diseño basadas en la separación de la "coordinación" del "cómputo".

En UWATc+ se brinda un diagrama de representación de contrato, donde se describen todos los datos que lo instancian. Cada tipo de dato y valor, pertenece a un elemento del meta-modelo de la figura 6.1. Teniendo en cuenta la figura ??, en primer lugar (item 1) se identifican los objetos participantes en el contrato; en el ejemplo de la figura ?? *DiscussionAction* y *UserAction* hacen referencias a dos clases reales perteneciente a la implementación de la herramienta Foro y

---

<sup>c</sup>CED: es el primer prototipo de una herramienta que implementa el uso de la coordinación de contrato en aplicaciones Java. La herramienta pertenece a ATX Software ([www.atxsoftware.com.ar](http://www.atxsoftware.com.ar)); fue desarrollada en Java y es de código abierto.

## 7.1. Usos de los contratos sensibles al contexto: Evidencias en el uso de entornos E-learning

Figura 7.5: Diagrama del contrato: Edición

Usuarios de la Aplicación Obra Abierta, respectivamente. Luego, se identifican los nombres de los parámetros context-aware significativos para el contrato, alineados en la misma columna del objeto que lo comparte (item 2). En Servicios (item 3) deben ser representados los métodos del objeto, que al ser ejecutados, provocan la intervención del contrato. Para este ejemplo *initState* y *getIdentifier* son ejecutados cuando un usuario ingresa a la herramienta Foro y las posteriores funcionalidades (servicios) disponibles dependen de la ejecución del contrato *Edición* (la figura ?? muestra la superposición del contrato entre los servicios de edición y las nuevas interfaces o funcionalidades). Las siguientes filas (items 4 y 5) se refieren a las pre y post-condiciones que se deben cumplir en la ejecución del contrato. Por último se explicitan las reglas de coordinación. Siguiendo con el ejemplo, en la parte del condicional *u.contexto = 'l1;p1;docente;r1;c1;'* verifica si el contexto del usuario *u* está compuesto por la locación *l1*, tienen el perfil *p1*, es un *docente*, cumple el rol *r1* y pertenece a la categoría *c1* (este tipo de representación de contexto se encuentra desarrollado en (57)). En cuanto a la acción de la regla de coordinación, continuando con el mismo ejemplo, se induce la ejecución del método *showMessage* del objeto *d* (DiscussionAction). El final del diagrama está dedicado a comentarios generales; cada comentario debe ir acompañado con el número de item (1,2,3,4,5 o 6) al que hace referencia.

Contrato: Edición			
1.	Participantes:	d:DiscussionAction	u:UserAction
2.	Param. c-a:	state, portlet, rundata, context	contextidentifier, identifier
3.	Servicios:	initState()	getIdentifier()
4.	Pre-Cond:	existe < contexto >	existe < contexto >
5.	Pos-Cond:	modifica < contexto >	
6.	Reglas de Coordinación:	Si u.contexto='p1;d;r1;c1;' entonces d.showMessage(data,string)	
Comentario			
1. DiscussionAction y UserAction pertenecen a clases implementadas en JAVA del proyecto Sakai. 4 y 5. < contexto > refiere a un objeto donde se oculta toda la información de contexto que caracteriza a los usuarios de la plataforma			

## 7.2. Desarrollos Context Aware en el campo del sonido

En el área de la música y del sonido, encontramos a nivel internacional, un buen número de desarrollos donde se ha aplicado la tecnología context-awareness, realizados en el marco de distintas universidades. Existen en la actualidad, varios sistemas context-aware para reproducir librerías de canciones teniendo en cuenta por ejemplo, un contexto de preferencias del usuario. Un tipo de estos desarrollos, entre los que se cuentan más de cincuenta, es el de la Universidad de Cambridge: [http://www.cl.cam.ac.uk/~mv253/suggested\\_projects\\_2004.html](http://www.cl.cam.ac.uk/~mv253/suggested_projects_2004.html) Dentro de la filosofía del software libre citamos a modo de ejemplo, un proyecto que reviste importancia para la investigación en Música de la Universidad de Victoria (Canadá) ([www.mistic.ece.uvic.ca/research](http://www.mistic.ece.uvic.ca/research))), destacamos también la serie de publicaciones recientes disponibles, en formato pdf, en el sitio de dicha Universidad. Asociado a esta temática es interesante consultar el sitio <http://www.musicir.org/evaluation/> que nos introduce en el objetivo del proyecto IMIRSEL (International Music Information Retrieval Systems Evaluation Laboratory) en el que se brindan recursos para el desarrollo y evaluación de técnicas científicas para la captura de información musical (MIR - Music Information Retrieval) y técnicas sobre el uso de librería de música digital (MDL - Music Digital Library). Parte del proyecto se relaciona a la creación de material musical de colecciones a gran escala, seguro y accesible; en variedades de audio y en forma de meta-datos. Estas colecciones, acopladas con un conjunto de tareas experimentales estandarizadas y métricas de evaluación estándares, harían posible la participación de los miembros de la comunidad de investigación internacional MIR/MDL en TREC (<http://trec.nist.gov/>) – como evaluación de la “competencia” ellos pueden comparar y contrastar científicamente sus métodos conformando un vasto almacenamiento de material musical disponible para el mundo. IMIRSEL se encuentra localizado en la Graduate School of Library and Information Science (GSLIS), en la universidad de Illinois en Urbana-Campaign (UIUC). Setephen Downie, es investigador principal del proyecto de GSLIS y el profesor Michael Welge es co-director, miembro del ALG (Automated Learning Group) del Centro Nacional para aplicaciones de supercómputo (NCSA - National Center for Supercomputing Applications).

Una hipótesis sobre sistemas context-aware dinámicos aplicados al campo audiovisual Si conceptualizamos a los sistemas context-aware como herramientas aptas para que un usuario pueda obtener información de forma eficiente y en concordancia con un determinado contexto, estamos ante un sistema que se determina a través de tres componentes fundamentales: 1) Un usuario con su contexto, pudiendo estar formado con información propia o del entorno que accede a... 2) Una segunda componente, donde efectivamente se implementan algoritmos, protocolos, semánticas, etc., brindando servicios con características contextaware. 3) Y,

## 7.2. Desarrollos Context Aware en el campo del sonido

---

la última componente integrada por los objetos que representan cualquier tipo de información y contenido que pueda ser útil para el usuario en cuestión. Este tipo de información puede ser representada de forma similar a los Learning Object Metadata (LOM)

d

del campo de los sistemas e-learning colaborativos, y referenciando al área audiovisual estos objetos podrían ser – entre otras cosas – abstracciones de señales digitales (imagen-sonido), fragmentos de videos, películas, bandas sonoras, obras musicales, letras, guiones literarios y cualquier tipo de elemento esencial para la composición, estudio y consulta requerido por el usuario.

Posibilidades de un usuario para obtener objetos de información.

La figura 7.2, caracteriza componentes de un modelo representativo de una familia de herramientas con características context-aware que fueron investigadas para el campo audiovisual, a las que podemos sumar a esta referencia la arquitectura RACOFI (Anderson et al. , 2003) para la composición de objetos, integrada por algoritmos de filtros colaborativos para la obtención de learning object de repositorios (a través del subsistema denominado SLOPE ONE) y reglas de inferencias para personalizar la selección de los objetos (a través del subsistema denominado RULEML). Otro tipo de arquitectura orientada a la recuperación y composición de información es la utilización de técnicas de la Web Semántica (Berners-Lee, 1998); como el caso de utilización de la Web Semántica para brindar servicios context-aware de recomendaciones de temas musicales (Elliott y Tomlinson , 2006). Retomando el análisis de la figura 5.2, principalmente en la segunda componente se encuentran representadas, entre otras técnicas y modelos, las reglas de inferencias; lo cual permite una posible sustitución por una componente contrato context-aware. De esta manera se obtendrá el nuevo modelo que representa la figura 5.3, que evoluciona desde los sistemas referenciados en el segundo bloque de la figura 5.1, hacia un dispositivo hipermedial context-aware dinámico, representado en el nivel del tercer bloque de 5.1.

---

<sup>d</sup>Learning Object Metadata (LOM, metadatos para objetos de aprendizaje) es un modelo de datos, usualmente codificado en XML, usado para describir un objeto de aprendizaje y otros recursos digitales similares usados para el apoyo al aprendizaje. Su propósito es ayudar a la reutilización de objetos de aprendizaje y facilitar su interaccionalidad, usualmente en el contexto de sistemas de aprendizaje on-line (LMS).

Entonces, las soluciones en el campo audiovisual referidas a sistemas context-aware pueden evolucionar con precisos fundamentos en el marco de los modelos genéricos referidos en la sección 5.3, hacia nuestra propuesta sobre los dispositivos hipermediales dinámicos; siempre y cuando sea posible tanto a nivel tecnológico como funcional, la inclusión de la componente contrato.

#### 7.2.1. Conclusión

En base a la experiencia recogida en el proyecto Obra Abierta, es posible asegurar que la implementación del modelo de diseño UWATc+ en el ciclo de vida del desarrollo Aplicaciones E-Learning permitió un mejor entendimiento entre los expertos en educación, diseñadores y programadores. El modelo también ayudó en la comprensión de la teoría de coordinación de contrato aplicada a transacciones para Aplicaciones e-learning.

### 7.3. DHD: Prototipo experimental

#### 7.4. Intervención del DHDCca en un modelo de simulación DEVS para DHD

##### 7.4.1. Construcción de métricas para el DHD

Es numerosa la información existente referida a la definición de métricas e indicadores, sin un claro consenso en cuanto a la terminología. En este sentido consideramos que la Ontología de Métricas e Indicadores presentada en [7] se constituye en una importante propuesta para el área de gestión calidad y un aporte valioso para las actividades implicadas en dicha gestión. Si bien hay estudios en el tema, en este trabajo nos enfocaremos en el marco de medición y evaluación orientado a propósitos, denominado INCAMI (Information Need, Concept model, Attribute, Metric and Indicator) [8]. INCAMI se fundamenta en el método Web-QEM (Web Quality Evaluation Method) [9], el cual se basa en modelos y métricas de calidad y se centra en la evaluación cuantitativa de características y atributos de entidades. De esta manera, INCAMI puede ser utilizado en el diseño de requerimientos no funcionales, en la selección de métricas para cuantificar los atributos de las entidades involucradas y en la interpretación de los valores correspondientes mediante indicadores.



## 7.4. Intervención del DHDCca en un modelo de simulación DEVS para DHD

La primera fase corresponde a la definición y especificación de requerimientos. Este módulo trata con la definición de la necesidad de información (es decir, el foco de la evaluación) y el diseño de los requerimientos no funcionales, que servirán como guías para las actividades posteriores de medición y evaluación. Tomamos como punto de partida la descripción del componente conceptual básico del DHD denominado Paquete Hipermedial (PH) [10], en el cual se requiere necesariamente dos sujetos que interactúen entre sí a través de alguna herramienta TIC, pudiéndose verificar algún cambio de contexto en al menos uno de los sujetos y la participación inicial de un tercero como determinante constitutiva y dinámica de la red. La estructura del PH como componente conceptual básico quedó definida según la siguiente figura:

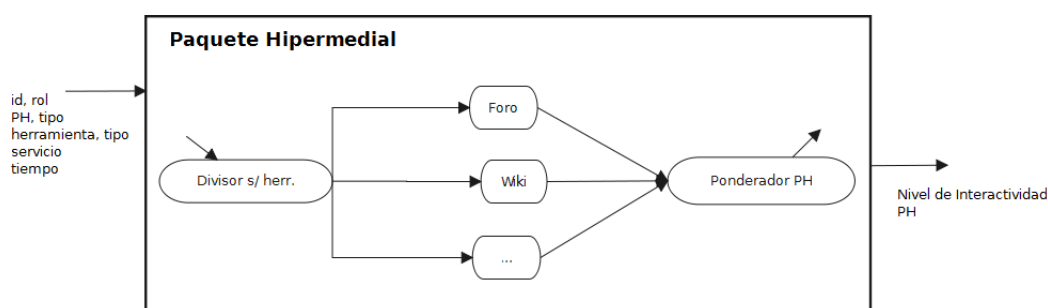
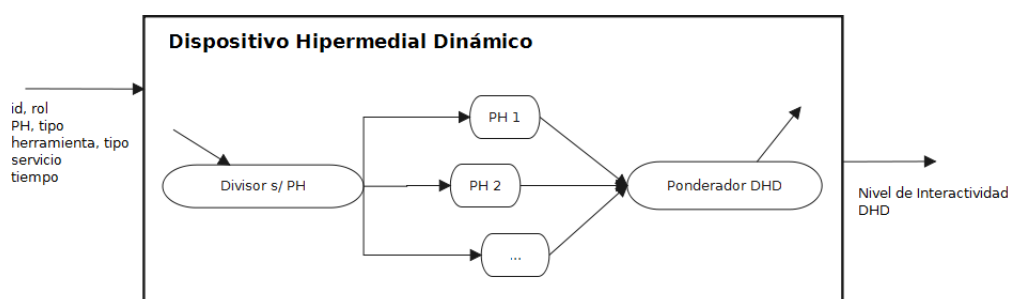


Fig. 1. Esquema de los módulos acoplados que se integran en un PH.

Y si pensamos que los Paquetes Hipermediales son los componentes conceptuales básicos del Dispositivo Hipermedial Dinámico debemos integrarlos para obtener el nivel total de interactividad de cada participación. La siguiente figura 2 nos muestra el esquema completo:



Esquema de los módulos acoplados que se integran en un DHD.

De esta manera se desprende que la información necesaria en nuestro caso es

función de las interacciones de los participantes, las cuales estarán definidas por: id del participante, rol del mismo, Paquete Hipermedial sobre el cual participa, tipo de PH, herramienta sobre la cual realiza la interacción, tipo de herramienta, servicio con el cual interactúa y el tiempo, día y hora de la interacción.

La fase siguiente corresponde al diseño e implementación de la medición. Este módulo trata con la definición de las métricas que serán útiles para cuantificar los atributos, que en la etapa anterior se identificaron como parte de la especificación de requerimientos, y que son de especial interés en el proyecto, dado que constituyen las características que se medirán para el ente a evaluar, considerando la necesidad de información establecida. Es decir, el objetivo final de la evaluación, que en nuestro caso es el nivel de interactividad de la participación.

De ambas figuras 1 y 2 entendemos la necesidad de plantear tres niveles de métricas para el análisis de las interacciones en tiempo real, dado que nos encontramos con tres entidades diferentes: la herramienta, el ponderador de los PH y el ponderador del DHD. Es fundamental comprender en el concepto de métrica, qué atributos se cuantifican y a qué entes los asociamos. Asimismo, es preciso identificar el tipo de valor que se obtiene, la unidad en la que se expresa y el tipo de escala que se usa, con el fin de poder realizar una apropiada interpretación y un análisis matemático y/o estadístico. Siguiendo con las recomendaciones del modelo INCAMI con el propósito de obtener valores para los indicadores globales, debemos tener en cuenta un modelo de acumulación y criterios de decisión. El modelo de ponderación y acumulación (agrupamiento) persigue la confección de un proceso de evaluación bien estructurado, objetivo y comprensivo para los evaluadores (o la evaluación en sí misma). Al igual que en otros casos de estudio [9], se usaron pesos, modelos de puntuación multi-criterio (consenso) para diseñar y ajustar procesos. Un modelo de ponderación (o puntuación) multi-criterio (o por consenso) como LSP, por Logic Scoring of Preference [11], en conjunción con propiedades de sincronización, neutralidad, reemplazabilidad y otras relaciones usando el agregado de operaciones basadas en el modelo matemático de pesos. El principal objetivo de la implementación de la evaluación global permite mayores niveles de flexibilización para los valores de los indicadores globales y parciales, a partir de los valores de indicadores elementales utilizando el modelo de agrupamiento obtenido para efectuar el cálculo. En este proceso, dichos valores deben ser acordados y consensuados por expertos con experiencia en el uso de este tipo de sistemas. A su vez mencionamos, que los valores numéricos indicados solo buscan mostrar un ejemplo de aplicación. Desarrollos posteriores posibilitarán el responsable de la evaluación pueda dar valor a los diversos coeficientes subrayando aquel atributo que considere más importante en el proceso. En cada caso, el valor resultado brinda una medida sobre el grado de interactividad de la participación.

## 7.4. Intervención del DHDCca en un modelo de simulación DEVS para DHD

---

### Métrica de la herramienta

Se construye por un producto de cuatro coeficientes:

Nivel de interactividad de la participación en la H =  $C1 * C2 * C3 * C4$

Esos cuatro coeficientes estarán en relación a:

Tipo de herramienta

De formato trasmisivo (ej.: links, recursos).  $C1 = 1$

De formato interactivo (ej.: foros, wiki).  $C1 = 2$

Tipo de servicio utilizado

Crear.  $C2 = 2$

Consultar.  $C2 = 1$

Editar.  $C2 = 2$

Borrar.  $C2 = 1$

Rol del participante

Docentes.  $C3 = 1$

Alumnos.  $C3 = 2$

Usuarios que utilizan la herramienta

Uno o dos participantes.  $C4 = 1$

Tres o más participantes.  $C4 = 2$

### Métrica del ponderador del PH

Se construye por un producto de tres coeficientes:

Nivel Interactividad de la participación en el PH =  $B1 * B2 * B3$

El valor de estos tres coeficientes serán:

Nivel Interactividad de la participación en la herramienta

$B1 = C1 * C2 * C3 * C4$

Tiempo entre la última participación y la actual

Si es menos de un día.  $B2 = 3$

Si es menos de una semana.  $B2 = 2$

Si es más de una semana.  $B2 = 1$

Intercalación entre las herramientas utilizadas

Si utiliza tres o más herramientas.  $B3 = 3$

Si utiliza dos.  $B3 = 2$

Si utiliza una.  $B3 = 1$

### Métrica del ponderador del DHD

## 7.4. Intervención del DHDCca en un modelo de simulación DEVS para DHD

---

La existencia de diversos tipos de PH (cursos y proyectos en entornos colaborativos)

De esta forma obtenemos:

Nivel Interactividad de la participación en el DHD =  $A1 \cdot A2$

El valor de estos dos coeficientes será:

Nivel Interactividad de la participación en el PH

$A1 = B1 \cdot B2 \cdot B3$

Tipo de Paquete Hipermedial

Si es un curso.  $A2 = 1$

Tomando otros valores según el tipo de PH.

Finalmente, en la etapa de evaluación, estas métricas deben ser interpretadas a través de indicadores con el objetivo de evaluar o estimar el grado de conformidad que los requerimientos propuestos alcanzaron. Es en este momento cuando deben seleccionarse los indicadores que interpretarán cada métrica que cuantifica a cada atributo correspondiente en el diseño de los requerimientos no funcionales. Los indicadores contienen también una escala y una función o algoritmo a través del cual será posible interpretar el valor de la métrica, con ayuda también de un criterio de decisión, que establecerá umbrales de aceptabilidad al valor obtenido.

### 7.4.2. Integración en el modelo DEVS

Continuando en la línea de la sección anterior, incorporamos las métricas en el modelo original DEVS, para potenciar el análisis de los procesos de educar, investigar, producir y gestionar, y posibilitar un posterior indicador para el cambio contextual de los participantes. De manera general un modelo DEVS atómico queda definido por la siguiente estructura:

$M = (X, Y, S, \text{int}, \text{ext}, \tau, \text{ta})$

donde:

X es el conjunto de valores de eventos de entrada.

Y es el conjunto de valores de eventos de salida.

S es el conjunto de valores de estado.

int, ext, y  $\tau$  son funciones que definen la dinámica del sistema.

En nuestro caso, el valor de  $\text{ta}$ , avance de tiempo, que señala el tiempo en

#### 7.4. Intervención del DHDCca en un modelo de simulación DEVS para DHD

---

que el sistema permanecerá en un estado determinado en ausencia de eventos de entrada, valdrá infinito. Por tanto, int, la función de transición interna, que recalcula el valor de estado del sistema cuando transcurre el tiempo  $t_a$ , no es necesario definirla.

Para el caso una herramienta genérica (Foro, Wiki, Blog, etc.): X: conjunto constituido por las interacciones de los participantes y es un vector de ocho componentes que incluye: id del participante, rol del mismo, Paquete Hipermedial sobre el cual participa, tipo de PH, herramienta sobre la cual realiza la interacción, tipo de herramienta, servicio con el cual interactúa y el tiempo, día y hora de la interacción.

Y: conjunto de valores que incluye (vector de nueve componentes): el nivel de interactividad en la herramienta, más el vector anterior.

S: estará determinado por el conjunto de valores de niveles de interactividad a partir de la métrica utilizada por la función de transición externa.

ext, función de transición externa: aquí es donde insertaremos la métrica para la herramienta expuesta en el apartado anterior. , función de salida: nos devolverá el valor de estado del sistema, es decir el nivel de interactividad de la participación analizada.

En la figura 1 vemos que el PH integra los modelos atómicos de las herramientas y suma a su vez dos módulos atómicos adicionales. El primero que denominaremos divisor PH, será el encargado de redireccionar los eventos a cada herramienta particular. Y el ponderador PH, el cual nos dará el valor global de interactividad del PH a partir del grado de interactividad de cada herramienta y la ponderación.

Para el divisor PH:

X: conjunto de valores de las interacciones de los participantes determinadas por (ocho componentes): id del participante, rol del mismo, Paquete Hipermedial sobre el cual participa, tipo de PH, herramienta sobre la cual realiza la interacción, tipo de herramienta, servicio con el cual interactúa y el tiempo, día y hora de la interacción.

Y: será el mismo evento de entrada, es decir el vector de ocho componentes, pero por el puerto correspondiente a la herramienta solicitada.

#### 7.4. Intervención del DHDCca en un modelo de simulación DEVS para DHD

---

S: corresponderá al valor de las componentes del vector del evento de entrada. ext, función de transición externa: tomará el valor del evento de entrada y a partir del valor del quinto componente definirá el puerto de salida.

, función de salida: devolverá el valor actual del estado del sistema, redireccionándolo por el puerto calculado en la función de transición externa. Para el ponderador PH:

X: el conjunto de valores de entrada coincidirá con los valores de salida de la herramienta (nueve componentes): nivel de interactividad de la participación en H, id del participante, rol del mismo, Paquete Hipermedial sobre el cual participa, tipo de PH, herramienta sobre la cual realiza la interacción, tipo de herramienta, servicio con el cual interactúa y el tiempo, día y hora de la interacción.

Y: nos devolverá en un vector de diez componentes: el nivel de interacción de la participación en el PH, más el evento de entrada.

S: estará determinado por el conjunto de valores de niveles de interactividad a partir de la métrica utilizada por la función de transición externa.

ext: tomará el valor del evento de entrada, nivel de interactividad de la herramienta y lo ponderará a partir de la métrica para el PH expuesta en el apartado anterior.

: devolverá el valor de estado calculado en la función de transición externa.

En la figura 2 vemos que el DHD integra los PH y suma a su vez dos módulos atómicos adicionales. El primero que denominaremos divisor DHD, será el encargado de redireccionar los eventos a cada PH perteneciente al DHD. Y el ponderador DHD, el cual nos dará el valor global de interactividad del DHD a partir del grado de interactividad de cada PH y la ponderación.

Para el divisor DHD:

X: conjunto de valores de las interacciones de los participantes determinadas por (ocho componentes): id del participante, rol del mismo, Paquete Hipermedial sobre el cual participa, tipo de PH, herramienta sobre la cual realiza la interacción, tipo de herramienta, servicio con el cual interactúa y el tiempo, día y hora de la interacción.

Y: será el mismo evento de entrada, es decir el vector de ocho componentes,

#### 7.4. Intervención del DHDCca en un modelo de simulación DEVS para DHD

---

pero por el puerto correspondiente al PH de interacción.

S: corresponderá al valor de las componentes del vector del evento de entrada. ext, función de transición externa: tomará el valor del evento de entrada y a partir del valor del tercer componente definirá el puerto de salida.

, función de salida: estará en función del valor actual del estado del sistema, redireccionándolo por el puerto calculado en la función de transición externa. Para el ponderador DHD:

X: el conjunto de valores de entrada coincidirá con los valores de salida del PH (diez componentes): nivel de interactividad de la participación en el PH, nivel de interactividad de la participación en H, id del participante, rol del mismo, Paquete Hipermedial sobre el cual participa, tipo de PH, herramienta sobre la cual realiza la interacción, tipo de herramienta, servicio con el cual interactúa y el tiempo, día y hora de la interacción.

Y: nos devolverá un vector de once componentes: nivel de interacción de la participación en el DHD, más el vector anterior.

S: estará determinado por el conjunto de valores de niveles de interactividad a partir de la métrica utilizada por la función de transición externa.

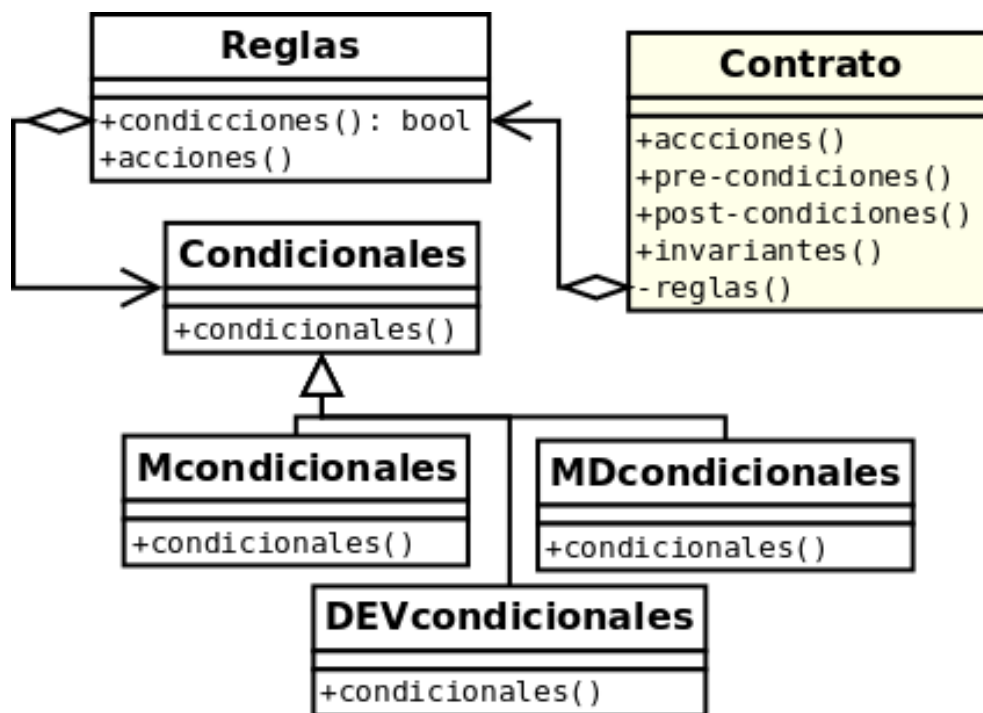
ext: tomará el valor del evento de entrada, nivel de interactividad del PH y lo ponderará a partir de la métrica para el DHD expuesta en el apartado anterior.

: devolverá el valor de estado calculado en la función de transición externa.

#### Condicionales para la coordinación de contratos

Ahora a través de un diagrama UML se definen las clases utilizadas en la implementación de los Condicionales DEVS dentro de las reglas de los contratos, donde se mantienen las propiedades e influencias (relaciones entre elementos conceptuales) descritas en la figura 1 .

La figura 2 describe los elementos y relaciones relevantes en la creación de condicionales inferidos por métricas de interacción (sección 4) implementadas en un modelo de simulación DEVS integrado (sección 3).



Partiendo de una de las propiedades de las reglas de los contratos sobre la posibilidad de definir comportamiento a través de parámetros context-aware (sección xxx) e inducidos por reglas donde se designan partes de las acciones del contrato. De esta manera, las reglas forman parte de un mecanismo de agregación encargado de la composición de diferentes tipos de condicionales, en los que se encuentran una familia de condicionales (Condicionales DEVS) conectados a los métodos que implementan las métricas definidas particularmente para la simulación de interacciones (sección xxxx). Las otras dos familias de condicionales representadas por las clases Mcondicionales y MDcondicionales se comportan de manera similares teniendo en cuenta el mismo modelo de integración propuesto [cacic 2007]

A continuación se describe los aspectos principales que se tuvieron en cuenta en la integración de los anteriores sistemas de coordinación de contratos sensibles al contexto y extensiones de condicional para la aplicación de un nuevo sistema de métricas de interacciones Sakai mediante un modelo de simulación DEVS.

### Modelo conceptual de integración

Para implementar la invocación de métricas mediante métodos correctos, propusimos desde la perspectiva del rediseño e implementación computacional, un modelo de integración de muy bajo costo, sin cambios sustanciales ni en la arquitectura original ni en el código de la implementación dentro de la aplicación

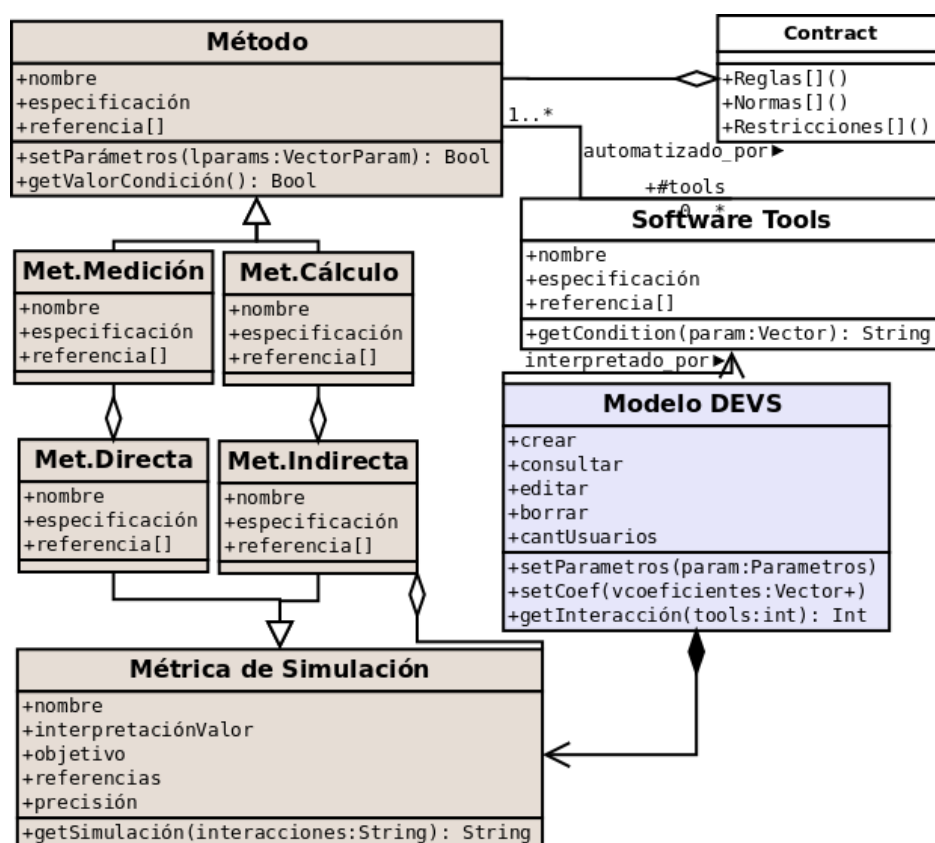


## 7.4. Intervención del DHDCca en un modelo de simulación DEVS para DHD

modificada en el proceso de inyección de las propiedades de coordinación de contratos sensibles al contexto [jornal de la clei].

El modelo conceptual de métrica pertenece al Modelo INCAMI (Information Need, Concept model, Attribute, Metric and Indicator: Información relevante, Modelo Conceptual, Atributos, Métricas e Indicadores) (Olsina, L., G. Lafuente y G. Rossi, 2000). INCAMI es un framework organizacional, orientado a la medición y evaluación que permite economizar consistentemente, no sólo metadata de métricas e indicadores, sino también valores mensurables en contextos físicos.

Por medio de un diagrama UML, se representa un modelo general de integración, teniendo en cuenta experiencias vinculadas al agregado de nuevas componentes en determinadas implementaciones resueltas para sistemas Elearning similares al diseño del framework Sakai (57). La integración se produce mediante la conexión de las reglas, a través de sus condicionales, con una métrica representada con un método. A su vez, la métrica es interpretada por un modelo DEVS diseñado para devolver valores de simulación (?).



En la figura 3 se puede observar lo correspondiente a cada una de las áreas mencionadas, representada con colores diferentes. Además, se muestra que la principal componente para lograr la integración está representada por la incorporación

de una relación de agregación entre la componente contrato y la entidad método. Los condicionales de las reglas de los contratos son invocados (mediante un método explícito relacionado con la noción de los Condicionales DEVS, por ejemplo, `getForum_theme`) por medio de un mecanismo de callback que permite la correcta invocación de la métrica.

... completar lo que falta ...

La integración de la métrica con el modelo DEV se produce mediante una vinculación de composición a través de una interfase que permite configurar parámetros de entradas (en este caso: crear, consultar, editar, borrar, `catUsuarios`) descrito en los bloques DEVS que representan las métricas implementadas con las interfases descritas por la clase Métricas de Simulación. El método `setCoef` permite establecer las ponderaciones de los coeficientes que intervienen en la métrica (sección 4); el método `setParametros` representa todos los parámetros necesarios en la configuración de las herramientas que interpretan modelos DEVS, en este trabajo se usó PowerDEVS [ref PowerDEV]. Luego, el método `getInteracción` se encarga de obtener un valor número entero representando en nivel de interacción relacionada al parámetro `tools` encargado de referenciar a una herramienta del framework Sakai (ej, Foro, Wiki, etc.) que a su vez es representada también con un número entero diferente para cada herramienta.

La interpretación y manipulación de los resultados de interacciones resueltos en el Modelo DEVS es manipulado por una herramienta de software representada por la clase Herramienta Sakai [herramienta Sakai]. A su vez, por medio de callback desde el Método de la métrica se obtienen valores para configurar el condicional definitivo para el contrato a través del método `getCondición` publicado por la Herramienta Sakai.

.... Hablar un poco sobre la herramienta de las coreanas. ....

### Implementación en un caso de uso

Atendiendo a lo expuesto, seguidamente se implementan lo explicado en el entorno PowerDEVS [12]. El caso de uso fue desarrollado utilizando como punto de partida los Registros de Actividad de Moodle correspondientes a la implementación de una asignatura de grado del Campus Virtual de la UNR [13].

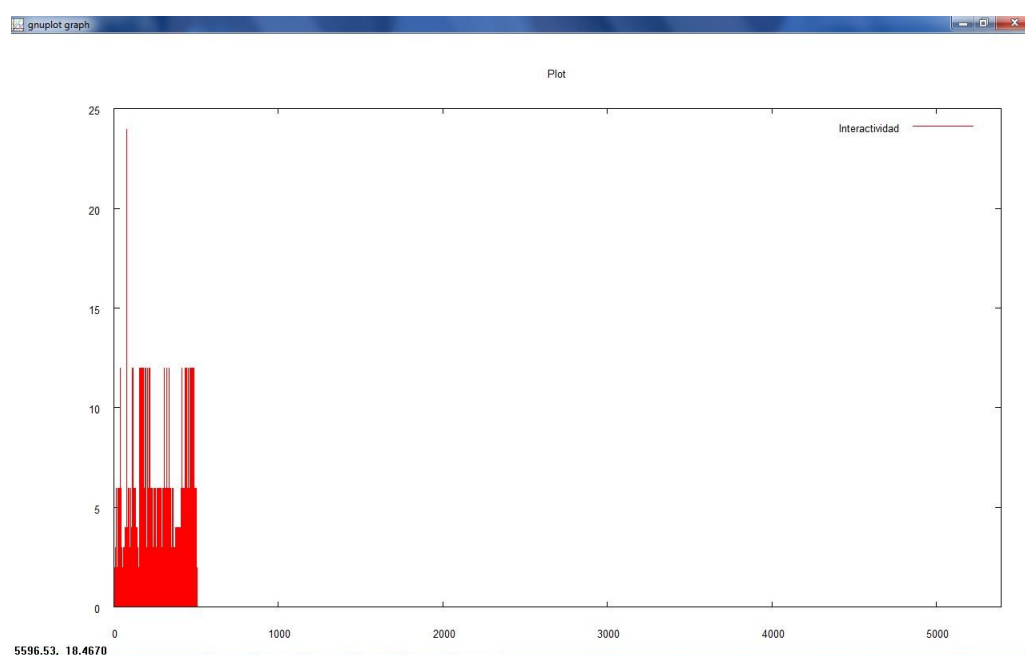
Cada métrica directa tiene asociado un método de medición claramente espe-

## 7.4. Intervención del DHDCca en un modelo de simulación DEVS para DHD

---

cificado. Las colecciones de datos son capturados desde la bases de datos (en este caso MySQL). Luego los datos son formateados según la necesidad explicitada en la sección anterior para posibilitar su lectura desde el entorno.

En la figura 3 comparamos los resultados obtenidos de Nivel de Interactividad para cada participación a través del tiempo, en los meses de Agosto/Septiembre (izquierda) y de Noviembre/Diciembre (derecha) para un curso seleccionado en el 2009.



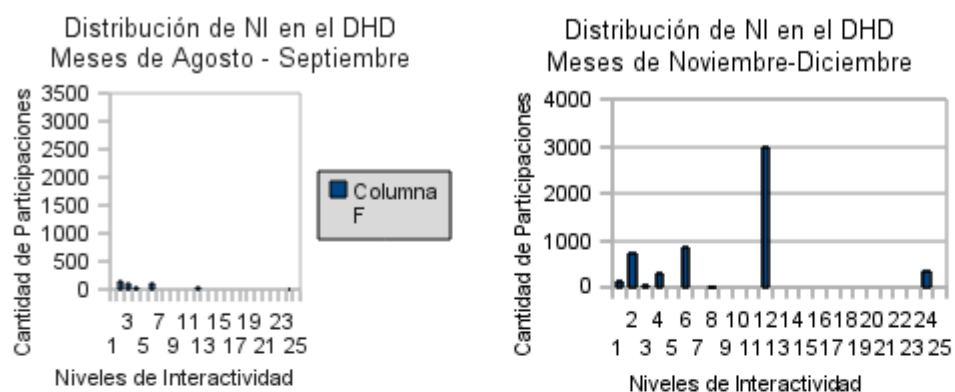
### Resultados obtenidos en el entorno PowerDEVS

Cabe mencionar que los resultados obtenidos son los globales del DHD, pero que sin embargo se disponen a su vez, los valores parciales tanto a nivel Paquete Hipermedial, como a nivel Herramienta individual, (por cuestiones de espacio no mostramos aquí dichos gráficos). Los mismos se exportan a un archivo que relaciona el número de participación, con su nivel de interactividad. Podemos ver en la Figura 4, el seguimiento global del curso mostrando la cantidad de interacciones totales dividida en cada nivel para los meses de Agosto-Septiembre y luego para los meses de Noviembre-Diciembre. De esta manera podemos observar como se va construyendo la modalidad participativa del DHD a través del crecimiento global de las interacciones (de 505 a 5394), teniendo como valor agregado más allá de la cantidad, las distribuciones de las mismas. Se verifica entonces en los dos últimos meses un 88 % de interacciones en niveles más participativos, en contraposición al 8



#### 7.4. Intervención del DHDCca en un modelo de simulación DEVS para DHD

---



### Postproceso de los resultados

Cabe mencionar que en el ejemplo presentado, los docentes propusieron estrategias didácticas específicas dentro del marco del Programa de Dispositivos Hipermediales Dinámicos que promovían la interactividad responsable para la construcción del conocimiento.

El resultado de este análisis podemos a su vez considerarlo una información de contexto, resignificando una característica del comportamiento de los participantes y atendiendo a la posibilidad de usar la información de interactividad como parámetro context-aware de los contratos [3]. Podremos entonces, establecer un lazo de retroalimentación entre las prácticas efectuadas en los entornos colaborativos, informadas en el Registro de Actividad y las acciones que devengan de los contratos.

### 7.4.3. Consideraciones generales

Fundamentados en el modelado sistémico del DHD, propusimos un primer desarrollo e implementación de métricas para el análisis completo de las interacciones a nivel Herramienta, a nivel Paquete Hipermedial y a nivel Dispositivo Hipermedial Dinámico. Este análisis tiene la versatilidad de estar directamente relacionado según los propósitos de importancia que determinen los sujetos responsables. De esta manera, brindan información calificada y aportan un camino de análisis evaluativo en tiempo real sobre cómo se desarrollan procesos de participación responsable a través de redes sociotécnicas para educar, investigar, gestionar y producir en el actual contexto físico-virtual.

Actualmente nuestros esfuerzos se enfocan al desarrollo de una herramienta de evaluación de espacios físicos-virtuales, integrada y de código abierto, que otorgue mayor flexibilidad y dinamismo a los componentes que pueden configurar al DHD para la construcción y disseminación de conocimiento.

#### **7.4. Intervención del DHDCca en un modelo de simulación DEVS para DHD**

---

Otras implementacions de los ContratosDHD

---

## Otras implementaciones de los ContratosDHD

---

### **8.1. Condicionales DEVS en la coordinación de contratos sensibles al contexto para los DHD**

#### **8.1.1. Introducción**

El actual contexto físico-virtual que se construye a partir de la utilización de las Tecnologías de la Información y Comunicación (TIC) posibilita a los sujetos ser partícipes de redes sociotécnicas conformadas por una multiplicidad de componentes y relaciones, que se configuran y reconfiguran por las diversas interacciones en función de una gran diversidad de requerimientos. En este sentido, el Programa interdisciplinario de I+D+T “Dispositivos Hipermediales Dinámicos” (DHD) [1], radicado en CIFASIS (CONICET-UNR-UPCAM), estudia la complejidad evidente de las mencionadas redes, integrando aportes de diversas disciplinas como informática, educación, ingeniería, psicología y antropología, entre otras.

Se conceptualiza como Dispositivo Hipermedial Dinámico -DHD- a la red heterogénea [2] conformada por la conjunción de tecnologías y aspectos sociales que posibilitan a los sujetos realizar acciones en interacción responsable con el otro para investigar, aprender, dialogar, confrontar, componer, evaluar, bajo la modalidad



### 8.1. Condicionales DEVS en la coordinación de contratos sensibles al contexto para los DHD

---

de taller físico-virtual, utilizando la potencialidad comunicacional, transformadora y abierta de lo hipermedial, regulados según el caso, por una “coordinación de contratos” [3]. En la implementación y optimización de un DHD para la producción y diseminación de conocimiento, los mecanismos de medición y evaluación suponen una de las actividades principales para el análisis y el aseguramiento de la calidad de lo que se desarrolla a través de los mismos.

Los procesos de medición son fundamentales dado que permiten cuantificar un conjunto de características deseadas acerca de un aspecto específico de algún ente en particular, proveyendo una visión más o menos detallada de su estado o condición. Por su parte, la evaluación interpreta los valores obtenidos en la medición. Para dichos procesos de medición y evaluación es necesario obtener datos cuantitativos, a partir de métricas de atributos de entes y la posterior interpretación de la medida a partir de indicadores [4]. Funcionalmente el DHD es conceptualizado como sistema complejo [5], en el cual los Participantes (P) a través del intercambio analítico y de producción de textos mediatizados en diversos tipos de formatos digitales, construyen las posibilidades y limitaciones de la mediación interdisciplinaria responsable en su área de incumbencia, siendo deseable que se pueda observar un paulatino cambio en su situación contextual. Al constatar que la característica primordial del DHD, es que las interacciones se deben a la ocurrencia asincrónica de eventos, hemos optado por el modelado con DEVS, Discrete Event System specification [6], considerándose además la gran adaptación del formalismo para modelizar sistemas complejos, y su simplicidad y eficiencia en la implementación de simulaciones.

Tecnológicamente el DHD está provista por un agregado de una pieza de software para la inyección de propiedades de coordinación de contratos sensibles al contexto [7]. Esta propiedad se logra a través de la implementación de contratos [8] con mecanismos de coordinación y componentes de sistemas context aware [9]. La utilización de reglas es parte esencial en la implementación de las acciones de los contratos y las tareas de coordinación. A su vez, las reglas están compuestas por los condicionales donde se centra parte de la lógica de adaptación que se requiere en los DHD. Algunos condicionales implementados requieren de mecanismos externos que colaboren en la composición de sus valores de verdad [10].

En este trabajo se define un nuevo tipo de condicional, denominado Condicional DEVS, para la inclusión de valores de verdad que puedan ser inferidos por medio de la implementación de un conjunto de métricas flexibles que contemplan las principales características de las interacciones de los participantes de los DHD. Tras esta introducción, en la sección 2 se identifican los elementos de los DHD en relación directa con los Condicionales DEVS. Luego, en la sección 3 se presenta un modelo de integración para el funcionamiento en una herramienta del framework

SAKAI (para la justificación de dicha elección ver [7]). En la sección 4 se describen algunas de las características principales de las métricas y un caso de uso concreto. Para finalizar, se presentan las conclusiones y consideraciones generales.

### 8.1.2. Aspecto tecnológico de los DHD

En esta sección se describen aspectos tecnológicos y componentes de los DHD que intervienen en el modelo de integración entre las métricas DEVS y el framework de nuestra propuesta. De esta manera, se solucionan los requerimientos sobre adaptación dinámica mediante la construcción de un modelo de contrato orientado a la implementación de servicios sensibles al contexto.

El uso de contratos parte de la noción de Programación por Contrato ("Programming by Contract") de Meyer [8] basada en la metáfora de que un elemento de un sistema de software colabora con otro, manteniendo obligaciones y beneficios mutuos. En nuestro dominio de aplicación consideraremos que un objeto cliente y un objeto servidor "acuerdan" a través de un contrato, -representado con un nuevo objeto-, que el objeto servidor satisfaga el pedido del cliente, y al mismo tiempo el cliente cumpla con las condiciones impuestas por el proveedor. A su vez las decisiones de comportamiento partirán de los condicionales de las acciones de los contratos.

Como ejemplo de la aplicación de la idea de Meyer en nuestro dominio de sistemas e-learning planteamos la situación en que un usuario (cliente) utiliza un servicio de edición de mensajes (servidor) a través de un contrato que garantizará las siguientes condiciones: el usuario debe poder editar aquellos mensajes que tiene autorización según su perfil (obligación del proveedor y beneficio del cliente); el proveedor debe tener acceso a la información del perfil del usuario (obligación del cliente y beneficio del proveedor). A partir de la conceptualización de contratos según Meyer se propone una extensión por medio del agregado de nuevas componentes para instrumentar mecanismos que permitan ejecutar acciones dependiendo del contexto. En aplicaciones sensibles al contexto [9], el contexto (o información de contexto) es definido como la información que puede ser usada para caracterizar la situación de una entidad más allá de los atributos que la definen. En nuestro caso, una entidad es un usuario (alumno, docente, etc.), lugar (aula, biblioteca, sala de consulta, etc.), recurso (impresora, fax, etc.), u objeto (examen, trabajo práctico, etc.) que se comunica con otra entidad a través del contrato.

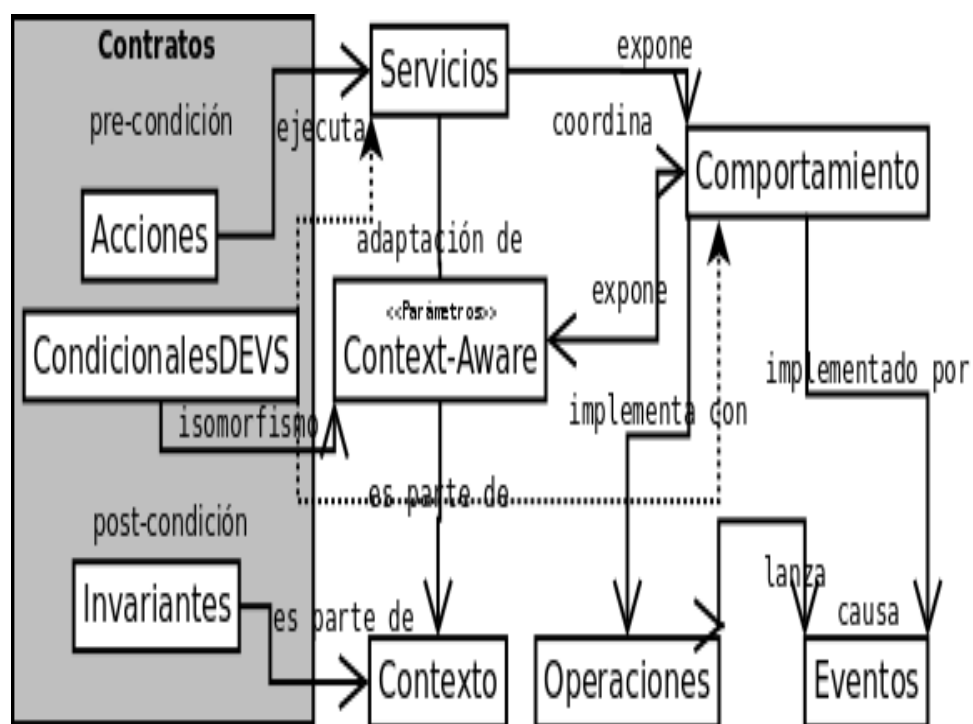
En [2] se propone una especificación del concepto de contexto partiendo de las consideraciones de Dourish [11] y adaptadas al dominio e-learning, que será la que

### 8.1. Condicionales DEVS en la coordinación de contratos sensibles al contexto para los DHD

consideraremos en este trabajo. Contexto es todo tipo de información que pueda ser censada y procesada, a través de la aplicación e-learning, que caracterizan a un usuario o entorno, por ejemplo: intervenciones en los foros, promedios de notas, habilidades, niveles de conocimientos, máquinas (direcciones ip) conectadas, nivel de intervención en los foros, cantidad de usuarios conectados, fechas y horarios, estadísticas sobre cursos, etc.

En términos generales, la coordinación de contratos es una conexión establecida entre un grupo de objetos influidas por condicionales que representan parte de la lógica de adaptación, aunque en este trabajo se consideran sólo dos objetos: un cliente y un servidor. Cuando un objeto cliente efectúa una llamada a un objeto servidor (ej., el servicio de edición de la herramienta Foro), el contrato “intercepta” la llamada y establece una nueva relación teniendo en cuenta el contexto del objeto cliente, el del objeto servidor, e información relevante adquirida y representada como contexto del entorno [1]. En este trabajo en los condicionales de las reglas se representarán diferente tipo de información de contexto con distinto grado de representación y abstracción, donde se requieren mecanismos de inferencias basados en la recolección, representación y simulación.

A continuación se brindarán detalles sobre algunas de los componentes y relaciones esenciales para la integración de este modelo con el framework utilizado y con los módulos que instrumentan la coordinación de contratos.



Modelo de elementos y relaciones de los DEVS condicionales.

Un contrato que siga las ideas de Meyer contiene toda la información sobre los servicios que utilizarán los clientes. Para incorporar sensibilidad al contexto nuestros contratos deberán tener referencias sobre algún tipo de información de contexto para su utilización. En el diagrama de relaciones entre entidades mostrado en la Figura 1 se describen los elementos que componen el concepto de contrato sensible al contexto donde se tiene participación de los condicionales DEVS. La figura comienza con la representación de un contrato según Meyer donde se caracterizan los principales elementos que lo componen (pre-condiciones, acciones, pos-condiciones). La flechas salientes de la zona gris indican los dos tipos de relaciones (acción-servicio e invariante-contexto) que se debe instrumentar para incorporar un mecanismo que provea a los contratos de la característica de sensibilidad al contexto. En la porción derecha de la Figura 1 aparecen las entidades necesarias para obtener contratos sensibles al contexto.

A continuación se explica cada uno de los elementos y su relación con los condicionales de las acciones.

- **Servicios:** En esta componente se representan los elementos necesarios para la identificación y clasificación de los servicios que pueden formar parte de las acciones de los contratos. Por ejemplo, nombre del servicio, identificadores, alcance, propósito, etc. En este caso existe una relación indirecta con el condicional DEVS establecida por la relación ejecutar entre la acción del contrato y el servicio.

- **Comportamiento:** El comportamiento de un servicio se logra a partir de combinar operaciones y eventos que son representadas con las componentes Operaciones y Eventos. De la misma manera el servicio puede ser implementado a través del uso de eventos, representados con el componente Eventos, que puede lanzar operaciones del componente Operaciones. Por ejemplo, de acuerdo con los roles (ej., alumno, instructor, docente, etc.) asignados a un usuario de una herramienta involucrado en un determinado contexto del entorno (ej., si está en un espacio Foro) y del usuario (ej., si tiene permiso de moderador), la componente Servicios brinda distintas funcionalidades (ej., editar un mensaje), que son instrumentadas por medio de operaciones concretas (ej., guardar un mensaje en una tabla) y/o a través de la publicación o subscripción de eventos. Aquí se establece una relación directa con el Condicional DEVS teniendo en cuenta todas las acciones que dependan de valoraciones influidas por el contexto, representadas por la simulación a través de un modelos DEVS.

- **Parámetros Context-Aware:** Se denomina Parámetros Context-Aware a la

### 8.1. Condicionales DEVS en la coordinación de contratos sensibles al contexto para los DHD

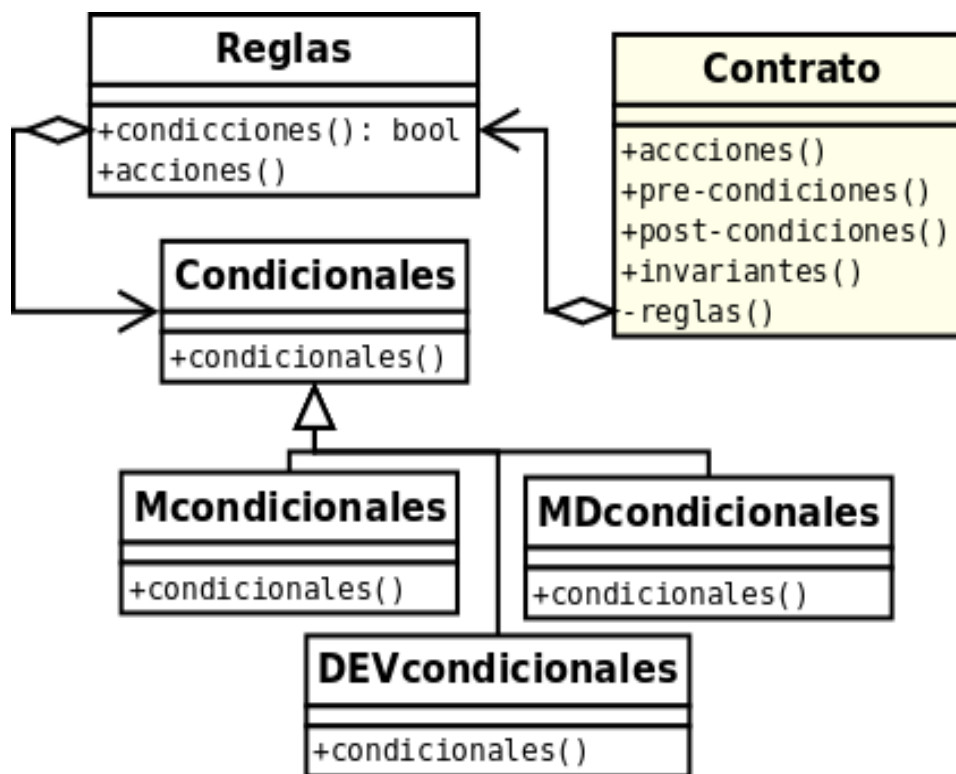
---

representación de la información de contexto que forma parte de los parámetros de entrada de las funciones y métodos exportados por los servicios, estableciendo de esta manera una relación entre el componente Servicios y el componente Parámetros Context-Aware. Existe una relación isomórfica entre los valores usados en los Condicionales DEVS y los elementos del conjunto de Parámetros Context-Aware.

- **Contexto:** Para nuestro modelo este tipo de información es utilizada de dos maneras diferentes: en primer lugar para la asignación de los valores que toman los Parámetros Context-Aware; en segundo lugar esta información puede ser utilizada para definir los invariantes que se representan en los contratos. Nuevamente se establece una relación indirecta entre los Condicionales DEVS y el contexto mediada por su representación como elemento de los Parámetros Context-Aware.

#### Condicionales para la coordinación de contratos

Ahora a través de un diagrama UML se definen las clases utilizadas en la implementación de los Condicionales DEVS dentro de las reglas de los contratos, donde se mantienen las propiedades e influencias (relaciones entre elementos conceptuales) descritas en la figura 1. La figura 2 describe los elementos y relaciones relevantes en la creación de condicionales inferidos por métricas de interacción (sección 4) implementadas en un modelo de simulación DEVS integrado (sección 3).



Elementos y relaciones relevantes en la creación de los condicionales

Partiendo de una de las propiedades de las reglas de los contratos sobre la posibilidad de definir comportamientos a través de parámetros context-aware e inducidos por reglas donde se designan partes de las acciones del contrato. De esta manera, las reglas forman parte de un mecanismo de agregación encargado de la composición de diferentes tipos de condicionales, en los que se encuentran una familia de condicionales (Condicionales DEVS) conectados a los métodos que implementan las métricas definidas particularmente para la simulación de interacciones (sección 4). Las otras dos familias de condicionales representadas por las clases **Mcondicionales** y **MDcondicionales** se comportan de manera similares teniendo en cuenta el mismo modelo de integración propuesto [10]. A continuación se describen los aspectos principales que se tuvieron en cuenta en la integración de los anteriores sistemas de coordinación de contratos sensibles al contexto y extensiones de condicionales para la aplicación de un nuevo sistema de métricas de interacciones mediante un modelo de simulación DEVS.

### 8.1.3. Modelo conceptual de integración

Para implementar la invocación de métricas mediante métodos correctos, propusimos desde la perspectiva del rediseño e implementación computacional, un

### 8.1. Condicionales DEVS en la coordinación de contratos sensibles al contexto para los DHD

---

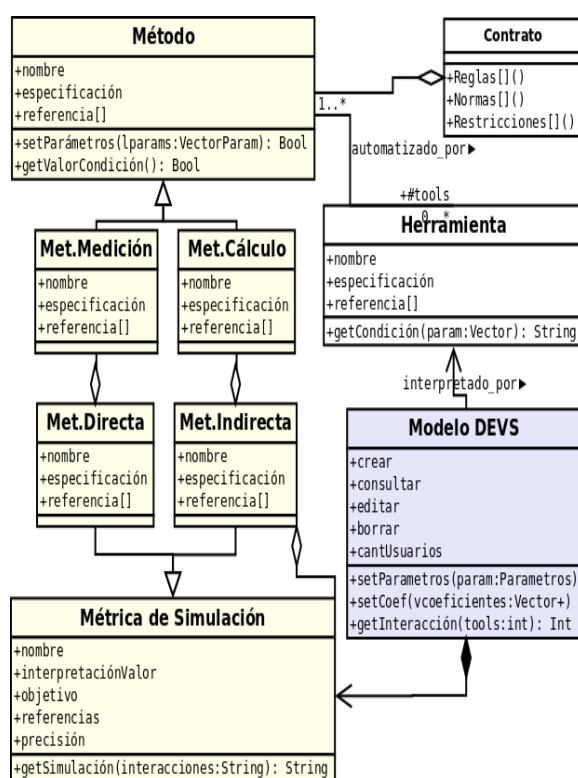
modelo de integración de muy bajo costo, sin cambios sustanciales ni en la arquitectura original ni en el código de la implementación dentro de la aplicación modificada en el proceso de inyección de las propiedades de coordinación de contratos sensibles al contexto [3].

El modelo conceptual de métrica pertenece al Modelo INCAMI (Information Need, Concept model, Attribute, Metric and Indicator: Información relevante, Modelo Conceptual, Atributos, Métricas e Indicadores) [12]. INCAMI es un framework organizacional, orientado a la medición y evaluación que permite economizar consistentemente, no sólo metadata de métricas e indicadores, sino también valores mensurables en contextos físicos.

Por medio de un diagrama UML, se representa un modelo general de integración, teniendo en cuenta experiencias vinculadas al agregado de nuevas componentes en determinadas implementaciones resueltas para sistemas e-learning similares al diseño del framework [2]. La integración se produce mediante la conexión de las reglas, a través de sus condicionales, con una métrica representada con un método. A su vez, la métrica es interpretada por un modelo DEVS diseñado para devolver valores de simulación [13].

En la figura 3 se puede observar lo correspondiente a cada una de las áreas mencionadas, representadas con colores diferentes. Además, se muestra que la principal componente para lograr la integración está representada por la incorporación de una relación de agregación entre la componente Contrato y la entidad Método. Los condicionales de las reglas de los contratos son invocados (mediante un método explícito relacionado con la noción de los Condicionales DEVS, por ejemplo, `getForumTheme`) por medio de un mecanismo de callback que permite la correcta invocación de la métrica. La primera fase de la misma corresponde a la definición y especificación de requerimientos. Este módulo trata con la definición de la necesidad de información (es decir, el foco de la evaluación) y el diseño de los requerimientos no funcionales, que servirán como guías para las actividades posteriores de medición y evaluación.

## 8.1. Condicionales DEVS en la coordinación de contratos sensibles al contexto para los DHD



Modelo de integración para contratos, métricas y modelo DEVS.

Tomamos como punto de partida la descripción del Dispositivo Hipermedial Dinámico [14]. De esta manera se desprende que la información necesaria en nuestro caso es función de las interacciones de los participantes, las cuales estarán definidas por: id del participante, rol del mismo, Paquete Hipermedial (PH) sobre el cual participa, tipo de PH, herramienta sobre la cual realiza la interacción, tipo de herramienta, servicio con el cual interactúa y el tiempo, día y hora de la interacción [13]. El principal objetivo de la implementación de la evaluación global permite mayores niveles de flexibilización para los valores de los indicadores globales y parciales, a partir de los valores de indicadores elementales utilizando el modelo de agrupamiento obtenido para efectuar el cálculo. En este proceso, dichos valores deben ser acordados y consensuados por expertos con experiencia en el uso de este tipo de sistemas. El seteo de los coeficientes serán establecido a través de la interface setcoeficient, de la clase Modelo DEVS (figura 3). En cada caso, el valor resultado brinda una medida sobre el grado de interactividad de la participación. El responsable de la evaluación pueda dar valor a los diversos coeficientes subrayando aquel atributo que considere más importante en el proceso. En cada caso, el valor resultado brinda una medida sobre el grado de interactividad de la participación. Este valor se obtiene a través de la interfase getInteracción, que toma como argumento en este caso un número entero que identifica la herramienta.



## 8.1. Condicionales DEVS en la coordinación de contratos sensibles al contexto para los DHD

---

Por último mencionamos que la interfase `setParametros` queda reservada para posibilitar diferentes relaciones algebraicas dentro de la métrica potenciando el nivel de expresión de la misma.

La interpretación y manipulación de los resultados de interacciones resueltos en el Modelo DEVS es manipulado por una herramienta representada por la clase `Herramienta`. A su vez la herramienta es la encargada de brindar la información necesaria sobre los parámetros que necesita la clase `Método` que es utilizada como argumento de la función `setParámetro`. El método `getValorCondición` representa los valores de verdad del condicional que formará parte de la regla explícita representada por el método `reglas` de la clase `Contrato`. Técnicamente la Herramienta es una aplicación que respeta la arquitectura del framework colaborativo SAKAI [7], utilizando los servicios base para el acceso a la base de datos. Por otro lado, permite la aplicación de una función transferencia que transforma dichos datos teniendo en cuenta un archivo de parametrización. En nuestros desarrollos se utilizó `iBatis` (<http://ibatis.apache.org/>) para el acceso a datos y `XML DOM` [<http://msdn.microsoft.com/en-us/library/ms764730>] la parametrización de la función transferencia. Los demás componentes tecnológicos que complementan el desarrollo cumplen los estándares del framework, en este caso se utilizaron `Servlets` y `Beans` teniendo en cuenta el acceso a los servicios base del framework que permitan el registro de la aplicación como herramienta.

### 8.1.4. Implementación en un caso de uso

A continuación se describe un caso de uso para ejemplificar las distintas etapas que se deben cumplimentar como usuario para la activación de las propiedades que brindan los Condicionales DEVS dentro de los contratos sensibles al contexto (sección 2). Además, se brindarán detalles funcionales sobre el uso de la herramienta Sakai que implementa la conexión entre los métodos de la métrica y el modelo de simulación DEVS, manteniendo la perspectiva de un usuario final. Se comienza con el diseño de las reglas de los contratos y sus correspondientes condicionales DEVS. Para esta etapa tomaremos como referencia el esquema para el diseño de contrato propuesto en UWATc en la última etapa del diseño de procesos e-learning Web [10].

### Representación de los Condicionales DEVS a través de UWATc

En UWATc se brinda un diagrama de representación de contrato, donde se describen todos los datos que lo instancian. Cada tipo de dato y valor, pertenece

## 8.1. Condicionales DEVS en la coordinación de contratos sensibles al contexto para los DHD

---

a un elemento del metamodelo de la figura 1.

Teniendo en cuenta el diagrama siguiente, en primer lugar (item 1) se identifican los objetos participantes en el contrato; en dicho ejemplo DiscussionAction y UserAction hacen referencia a dos clases reales perteneciente a la implementación de la herramienta Foro y Usuarios de una aplicación, respectivamente. Luego, se identifican los nombres de los parámetros context-aware significativos para el contrato, alineados en la misma columna del objeto que lo comparte (item 2). En Servicios (item 2) deben ser representados los métodos del objeto, que al ser ejecutados, provocan la intervención del contrato. Para este ejemplo initState y getIdentifier son ejecutados cuando un usuario ingresa a la herramienta Foro y las posteriores funcionalidades (servicios) disponibles dependen de la ejecución del contrato Edición. Las siguientes filas (item 2) se refieren a las pre y post-condiciones que se deben cumplir en la ejecución del contrato.

Por último se explicitan las reglas de coordinación (item 3). Siguiendo con el ejemplo, en la parte del condicional `u.contexto = ' l1; p1; docente; r1; c1; IT1; IPH1'` verifica si el contexto del usuario `u` está compuesto por la locación `l1`, tienen el perfil `p1`, es un docente, cumple el rol `r1` y pertenece a la categoría `c1` (este tipo de representación de contexto se encuentra desarrollado en [2]). Además, los últimos dos valores `IT1` y `IPH1`, representan el nivel de interacciones que un determinado usuario y de una herramienta particular, respectivamente. A diferencia de los otros valores, `IT1` y `IPH1` serán comparados con la resultante de la aplicación de una métrica de interacción a través de un elemento externo con interfaz para comunicarse a través de los Condicionales DEVS.

## 8.1. Condicionales DEVS en la coordinación de contratos sensibles al contexto para los DHD

Contrato: Edición			
1	Participantes:	d:DiscussionAction	u:UserAction
2	Param. c-a: Servicios: Pre-Cond: Pos-Cond:	state, portlet, rundata,context initState() existe < contexto > modifica < contexto >	contextidentifier, identifier getIdentifier() existe < contexto >
3	Reglas:	<b>Si</b> u.contexto='p1;d;r1;c1;IT1,IPH1' <b>entonces</b> d.showMessage(data,string)	
Condicionales			
4	DEVS	Valores	Coeficientes según métrica
		IT1	C1..4: valores de servicios B1..4: valores de las herramientas , donde $B_i = C1 * C2 * C3 * C4$
		IPH1	PH: valor para el PH, donde $PH = B1 * B2 * B3 * B4$
5	MD	-	-
6	M	-	-
7	Comentario	<b>Del Item 4:</b>  Especificación de las Métricas: Herramienta - PH - DHD [13].	

Modelo de integración para contratos, métricas y modelo DEVS.

En el ítem 4 se describe la forma de representar en el diagrama que dichos valores pertenecen al tipo de Condicional DEVS y la definición de los coeficientes que formarán parte de la métrica y serán alcanzado por los valores de los parámetros de las interfaces de los métodos que la representan. En cuanto a la acción de la regla de coordinación, continuando con el mismo ejemplo, en el ítem 3 se induce la ejecución del método showMessage del objeto d (DiscussionAction). El final del diagrama está dedicado a comentarios generales; cada comentario debe ir acompañado con el número al que hace referencia.

De esta manera, a través de la extensión del diagrama de contrato de la etapa 4 en el proceso de diseño de los Pe-Irn [10] se logra describir los principales componentes que se tienen en cuenta en el diseño e implementación de los Condicionales DEVS para los casos de uso similares al presentado.

### Ejemplo de ejecución de la métrica en PowerDEVS

Atendiendo a lo expuesto, seguidamente se implementan lo explicado en el entorno PowerDEVS [15] teniendo en cuenta el mismo caso de uso introducido en esta sección. En la figura 3, la clase Modelo DEVS contiene las interfaces que

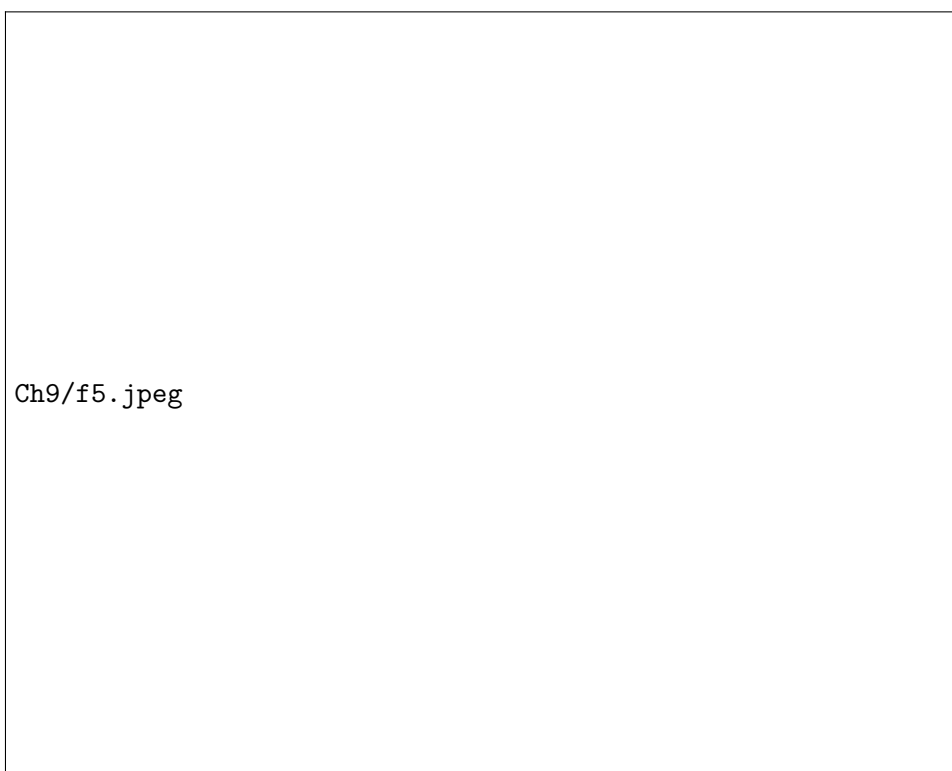
### 8.1. Condicionales DEVS en la coordinación de contratos sensibles al contexto para los DHD

---

son implementadas a través de la herramienta PowerDEVS para la interpretación de las métricas presentes en el comentario del diagrama de contrato.

Cada métrica directa tiene asociado un método de medición claramente especificado. Las colecciones de datos son capturados desde la bases de datos (en este caso MySQL). Luego los datos son formateados para posibilitar su lectura desde el entorno.

En la figura 4 se muestran, a manera de ejemplo, los resultados obtenidos de Nivel de Interactividad para cada participación a través del tiempo, en los meses de Noviembre-Diciembre para un curso seleccionado en el 2009, del Campus Virtual de la Universidad Nacional de Rosario (<http://www.campusvirtualunr.edu.ar>).



Resultados obtenidos en el entorno PowerDEVS.

Cabe mencionar que los resultados obtenidos son los globales del DHD, pero que sin embargo se disponen a su vez, los valores parciales tanto a nivel Paquete Hipermedial, como a nivel Herramienta individual, (por cuestiones de espacio nomostremos aquí dichos gráficos). Los mismos se exportan a un archivo que relaciona el número de participación, con su nivel de interactividad.

El resultado de este análisis es considerado una información de contexto, resig-

## **8.2. SEPI: una herramienta para el Seguimiento y Evaluación de Procesos Interactivos del DHD**

---

nificando una característica del comportamiento de los participantes y atendiendo a la posibilidad de usar la información de interactividad como parámetro context-aware de los contratos [3]. Podremos entonces, establecer un lazo de retroalimentación entre las prácticas efectuadas en los entornos colaborativos, informadas en el Registro de Actividad y las acciones que devengan de los contratos.

### **8.1.5. Conclusiones**

En este trabajo se fundamentó la posibilidad de extender las propiedades expresivas de las reglas de coordinación de contratos de los DHD a partir de los resultados de un mecanismo externo. La propuesta de integración expuesta sigue respetando las líneas de diseño e implementación establecidas por el modelo de los DHD. Ahora se tiene un nuevo mecanismo para la escritura de las reglas de los contratos que permitirá ahorrar esfuerzo en el diseño de los condicionales que verifiquen información de contexto parametrizado. Además, se evita la necesidad de tener conocimiento sobre las leyes (diseño) de las métricas y su implementación, con sólo tener información sobre la definición de los coeficientes que participan en la métrica.

## **8.2. SEPI: una herramienta para el Seguimiento y Evaluación de Procesos Interactivos del DHD**

### **8.2.1. Introducción**

Las actuales Tecnologías de la Información y Comunicación (TIC) han posibilitado la construcción de una nueva realidad físico-virtual donde los sujetos pueden ser partícipes de diversas redes sociotécnicas. Estas redes están conformadas por una multiplicidad de componentes y relaciones, que se configuran y reconfiguran por las diversas interacciones mediatizadas, en función de una gran diversidad de requerimientos.

La necesidad de evaluar y propiciar una mejor calidad de los procesos interactivos, cuyos propósitos se centren en investigar, educar y/o producir a partir de la participación responsable en contextos físicos-virtuales, adquiere especial relevancia para la construcción conjunta e inclusiva de la denominada “Sociedad de la Información y del Conocimiento”. En atención a esta necesidad insoslayable, el presente trabajo describe tecnológicamente el primer prototipo de una herramienta

integrada denominada “SEPI-DHD” que colabora en el Seguimiento y Evaluación analítica de los mencionados procesos de interacción mediatizados por un Dispositivo Hipermedial Dinámico (DHD) retroalimentando información de contexto de los participantes.

Se conceptualiza como DHD a una red sociotécnica [1] conformada por la conjunción de tecnologías y aspectos sociales que posibilita a los sujetos realizar con el otro, acciones en interacción responsable para investigar, aprender, dialogar, confrontar, componer, evaluar, diseminar bajo la modalidad de taller físico-virtual, utilizando la potencialidad comunicacional, transformadora y abierta de lo hipermedial [2], reguladas según el caso por una “coordinación de contratos” [3].

De esta manera, el DHD se constituye como una entidad compleja [4] compuesta por la integración de dos dimensiones indisociables: una técnica (o conjunto de técnicas constructivas que comportan una materialidad y una configuración particular) y una social dada por las relaciones intersubjetivas y la situación en la que se inscriben.

La necesidad de SEPI-DHD, se fundamenta en el marco de dos requerimientos de I+D relacionados: el primero está referido a la importancia de efectuar análisis evaluativos de los desarrollos e implementaciones llevados adelante en el entorno colaborativo de I+D+T del Programa “Dispositivos Hipermediales Dinámicos” [5]. El segundo requerimiento, se encuentra en una fase experimental e investigativa y está relacionado con la implementación de un original desarrollo de pieza de software denominada “contrato” que debe ser ubicada reflexivamente en el sistema por los usuarios calificados para diseñar e implementar el espacio de formación, investigación y/o transferencia con la finalidad de potenciar el aspecto dinámico de los procesos de interacción responsable del DHD [6].

En referencia a dichos requerimientos, adoptó la utilización del formalismo DEVS (Discrete EVents dynamic Systems) [7] que propone una teoría de modelado de eventos discretos en sistemas a tiempo continuo, permitiendo a su vez, una descripción modular de los fenómenos y el abordaje de la complejidad usando una aproximación jerárquica. En la implementación de dicho formalismo se integran las métricas de ponderación siguiendo las recomendaciones del framework INCAMI [8].

A su vez, tecnológicamente el entorno colaborativo de prueba está provisto por un agregado de una pieza de software para la inyección de propiedades de coordinación de contratos sensibles al contexto [9]. Esta propiedad se logra a través de la implementación de contratos [10] con mecanismos de coordinación y

## 8.2. SEPI: una herramienta para el Seguimiento y Evaluación de Procesos Interactivos del DHD

---

componentes de sistemas Context-Aware. La utilización de reglas es esencial para la implementación de las acciones de los contratos estando compuestas por los condicionales, en donde se centra parte de la lógica de adaptación que requiere el DHD. Algunos condicionales implementados requieren de mecanismos externos que colaboren en la composición de sus valores de verdad que son ponderados a partir de simulaciones por medio del modelo DEVS [11]. La herramienta SEPI-DHD de código abierto funcionalmente logra la integración tecnológica de los Condicionales DEVS, pertenecientes a los contratos con el módulo de ejecución del modelo DEVS [12].

En la siguiente sección describiremos los elementos tecnológicos del DHD en relación directa con los Condicionales DEVS. Luego, en la sección 3 presentaremos el modelo de integración adoptado para la implementación de la herramienta. En la sección 4 abordaremos algunas de las características, funcionalidades y un caso de uso concreto para arribar finalmente a breves conclusiones y prospectiva del desarrollo.

### 8.2.2. Elementos para la integración

En esta sección describiremos aspectos tecnológicos y componentes del DHD que la herramienta SEPI toma en cuenta para la conexión entre los Condicionales DEVS y las métricas de interacción DEVS.

Sintéticamente, los Condicionales DEVS son condicionales simples que pertenecen a un conjunto de reglas que definen acciones dentro de componentes computacionales contratos. Los contratos, definidos por Meyer [10], se basan en la metáfora de que un elemento de un sistema de software colabora con otro, manteniendo obligaciones y beneficios mutuos. En este caso, los contratos son objetos de primera clase dentro de la original propuesta tecnológica del DHD [6] donde se le agregan propiedades de sensibilidad al contexto y dinamismo. Dado que la evaluación sobre los valores de verdad de los Condicionales DEVS se referencian en resultados obtenidos al ejecutarse un simulador de un modelo de eventos discretos que representa una métrica cuali-cuantitativa [12], podemos afirmar que estos condicionales promueven nuevas propiedades de adaptación del DHD posibilitando que las reglas de los contratos tengan mayor grado de expresión.

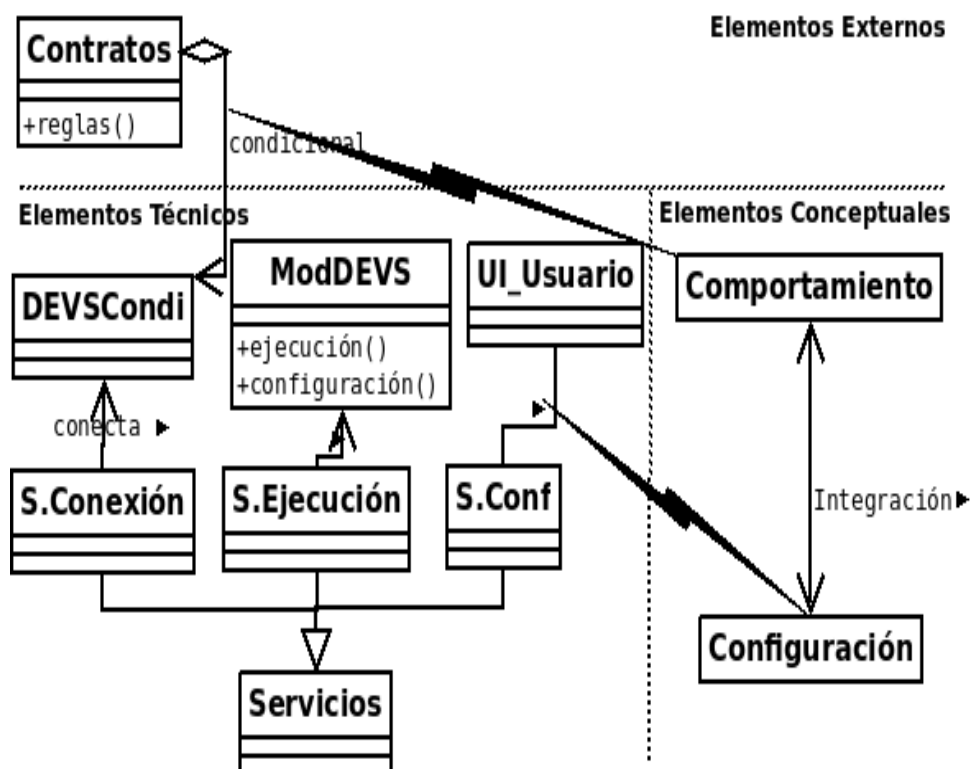
En la siguiente Figura 1, se muestra cuales son los elementos que se deben tener en cuenta para implementar los valores calculados que deben contrastarse con valores testigos para formar los condicionales de las proposiciones lógicas de las reglas. Las relaciones entre las componentes influyen en elementos conceptuales

## 8.2. SEPI: una herramienta para el Seguimiento y Evaluación de Procesos Interactivos del DHD

que determinarán algunas de las características funcionales de los requerimientos del DHD [9] relacionados a los requisitos de comportamiento y configuración.

La herramienta cuenta con tres tipos de servicios que derivan del núcleo base del framework del DHD, uno de ellos (S.Conexión) encargado de establecer la conexión para incorporar valores calculados dentro de los Condicionales DEVS. Otro tipo de servicio (S.Ejecución) implementa las secuencias de ejecuciones del modelo DEVS que obtiene los resultados de las interacciones. Algunos de estos pasos serán descritos en la sección 4 y forman parte de los requisitos que se deben cumplir para ejecutar la simulación implementada en el módulo DEVS (ModDEVS), que en este caso se ajusta a las interfaces que provee PowerDEVS [13].

El último servicio que extiende los servicios base (S.Configuración) se encarga de implementar todas las cuestiones de configuración de la métrica que provienen de la interfaz de usuario. En este caso se provee al usuario de un prototipo de interfaz que permite establecer las ponderaciones de los coeficientes de la métrica de interacción representada en DEVS, para que luego se puedan transformar en los parámetros necesarios al invocar el método de configuración de S.Configuración. El componente Configuración refiere a la relación entre el servicio de interfaz (S.Conf) y los elementos utilizados para implementar la interfaz del usuario de la herramienta, en este caso representada con la clase UIUsuario.





## 8.2. SEPI: una herramienta para el Seguimiento y Evaluación de Procesos Interactivos del DHD

---

Modelo de elementos y relaciones de los DEVS condicionales.

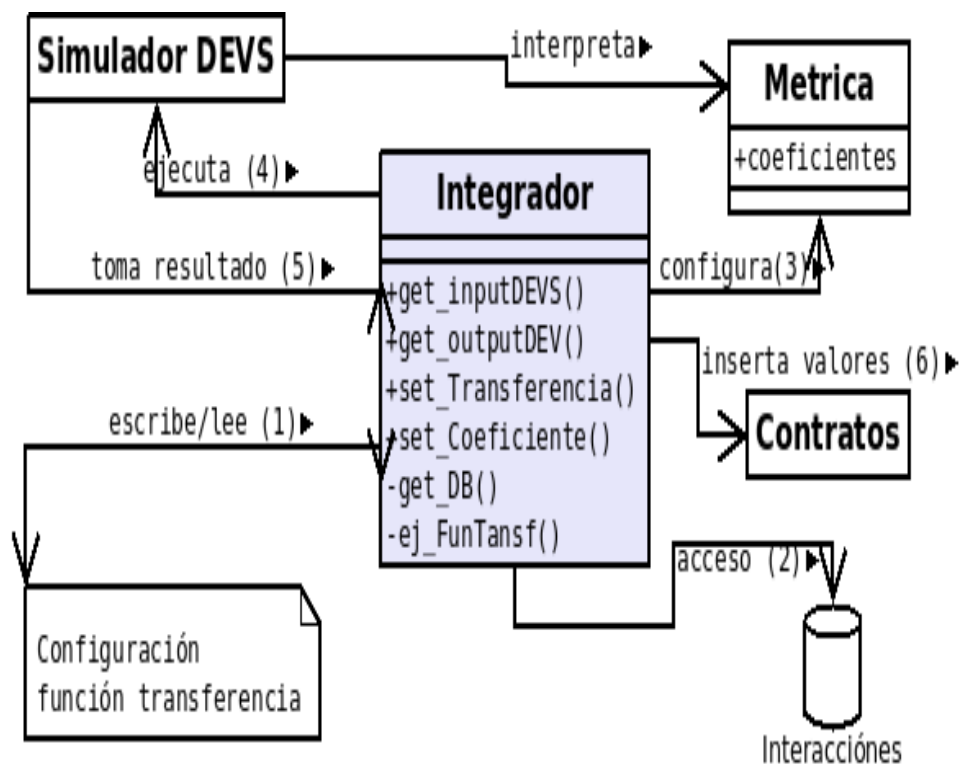
De esta manera queda establecida una relación conceptual entre los servicios (Servicios) y los contratos que definen el comportamiento adaptativo de algunas de las acciones del DHD. Esta última relación fue representada en la Figura 1 mediante la relación de agregación entre el contrato (Contrato) y el Condicional DEVS (DEVSCond), determinada por la relación conceptual Comportamiento. Las interfaces de los métodos las abordaremos seguidamente como parte de la herramienta que implementa el modelo de integración. De esta manera queda establecida una nueva relación conceptual, llamada relación de Integración, a partir de las componentes conceptuales Comportamiento y Configuración.

Cabe destacar que los elementos conceptuales representados en la figura forman parte de los requerimientos de adaptabilidad que debe cumplir el DHD [6]. A su vez, los elementos técnicos pertenecen al diseño propuesto para el modelo de integración que se implementa a través de la herramienta. Las funcionalidades que interpretan el servicio de ejecución las describiremos en la sección 4.

### 8.2.3. Modelo conceptual de integración

En esta sección representaremos un modelo general de integración teniendo en cuenta experiencias vinculadas al agregado de nuevas componentes en determinadas implementaciones resueltas para entornos colaborativos similares al diseño del framework [6]. En este caso, el modelo determina los componentes, sub-sistemas y relaciones sobre la herramienta de integración encargada de articular los resultados de la simulación del modelo DEVS para insertarlo en los contratos.

En la siguiente Figura 2 observamos una primera conexión entre los servicios de la herramienta, que fueron introducidos en la sección anterior, con el modelo de simulación DEVS. Para este caso contamos con los métodos del módulo Integrador, representado por una clase UML [14] que implementan las secuencias de ejecuciones que se deben respetar para que el intérprete DEVS (en este caso PoweDEVS [13]) tome los valores de entrada adecuados (método getInputDEVS) que son procesados por medio de una función transferencia parametrizada (método setTransferencia) a partir de un archivo de configuración XML. Luego se toman los valores de salida (método getOutputDEVS) para su posterior procesamiento. Todos los datos de entradas son extraídos de la base de datos perteneciente al entorno colaborativo utilizado, a través de métodos privado getDB. Los parámetros que implican la ejecución del simulador DEVS deben ser alcanzados por las interfaces representadas en la clase Integrador.



Modelo de integración para contratos, métricas y modelo DEVS.

Las cuestiones de parametrización y representación de los valores, pre y pos-condiciones que se encuadren en un nivel superior para lograr una mejor interpretación y manejo por parte de los usuarios se guardan en el archivo de configuración e interviene el servicio S.Configuración mencionado en la sección 2.

Si bien, en este caso, el diseño de integración propuesto se basa en la descripción de componentes de software de la herramienta de integración aquí propuesta, es necesario tener en cuenta otros elementos que en este diseño no se muestran. En otro trabajo [15] hemos desarrollado lo correspondiente a cada una de las áreas mencionadas con un nivel más profundo de detalle. Además, se muestra que la principal componente para lograr la integración está representada por la incorporación de una relación de agregación entre una componente Contrato y una entidad Método que se corresponde con la arquitectura que propone INCAMI [8] para la construcción de métricas.

En términos generales, la SEPI-DHD es una aplicación que respeta la arquitectura del framework, utilizando los servicios base para el acceso a la base de datos. Por otro lado, permite la aplicación de una función transferencia que transforma dichos datos teniendo en cuenta un archivo de parametrización. Los diversos componentes tecnológicos que complementan el desarrollo cumplen los estándares del

## 8.2. SEPI: una herramienta para el Seguimiento y Evaluación de Procesos Interactivos del DHD

---

framework, en este caso utilizamos Servlets y Beans teniendo en cuenta el acceso a los servicios base del framework que permitan el registro de la aplicación como herramienta.

### 8.2.4. Implementación en un caso de uso

A continuación desarrollaremos un caso de uso brindando detalles funcionales sobre la herramienta desde la perspectiva de un usuario final. Dicho camino se divide en seis pasos metodológicos coincidentes con las relaciones descritas en la Figura 2.

**Paso 1: Escritura y lectura del XML** La herramienta contiene una interfaz apropiada para la representación de los coeficientes que componen la métrica, permitiendo de esta manera que el usuario controle las ponderaciones que reflejan propiedades cuali-cuantitativas sobre las interacciones de los participantes. Se presentan los pesos necesarios (Rol, Tipo de Paquete Hipermedial, Tipo de Herramienta, Tipo de Servicio) con un valor por defecto igual a 1 (para los valores nulos el valor por defecto es cero). A su vez se da la posibilidad de modificar esos valores a través de la opción Modificar Ponderación, seleccionando el coeficiente y el subtipo de coeficiente. Luego se crea automáticamente un archivo XML donde se detallan todos los valores necesarios para que la función transferencia procese las tablas y valores de sus campos que sirvan como entradas al simulador DEVS. La siguiente tabla describe el significado de las marcas ("tags") donde se brindan los detalles concretos para los parámetros de las interfases de cada subsistema de la sección 3.

<code>&lt;dhdDB&gt; Sakai &lt;/dhdDB&gt;</code>	Nombre de la base de datos del sistema colaborativo
<code>&lt;dhdTable&gt; sakai_event &lt;/dhdTable&gt;</code>	Tabla base de datos Sakai
<code>&lt;dhdField&gt; EVENT &lt;/dhdField&gt;</code>	Campo de la base de datos
<code>&lt;dhdConditionType&gt; igual &lt;/dhdConditionType&gt;</code>	Tipo de comparación
<code>&lt;dhdValue&gt; content.new &lt;/dhdValue&gt;</code>	Valor del campo para comparar
<code>&lt;dhdValueTrue&gt; 1 &lt;/dhdValueTrue&gt;</code>	Valor que se devuelve la función transferencia si la condición es verdadera
<code>&lt;dhdValueFalse&gt; nothing &lt;/dhdValueFalse&gt;</code>	Valor que se devuelve la función transferencia si la condición es falsa
<code>&lt;dhdValueFalse&gt; DEVS.c1.DEVS.c2 &lt;/dhdValueFalse&gt;</code>	Referencia el coeficiente de ponderación de la métrica representada en DEVS

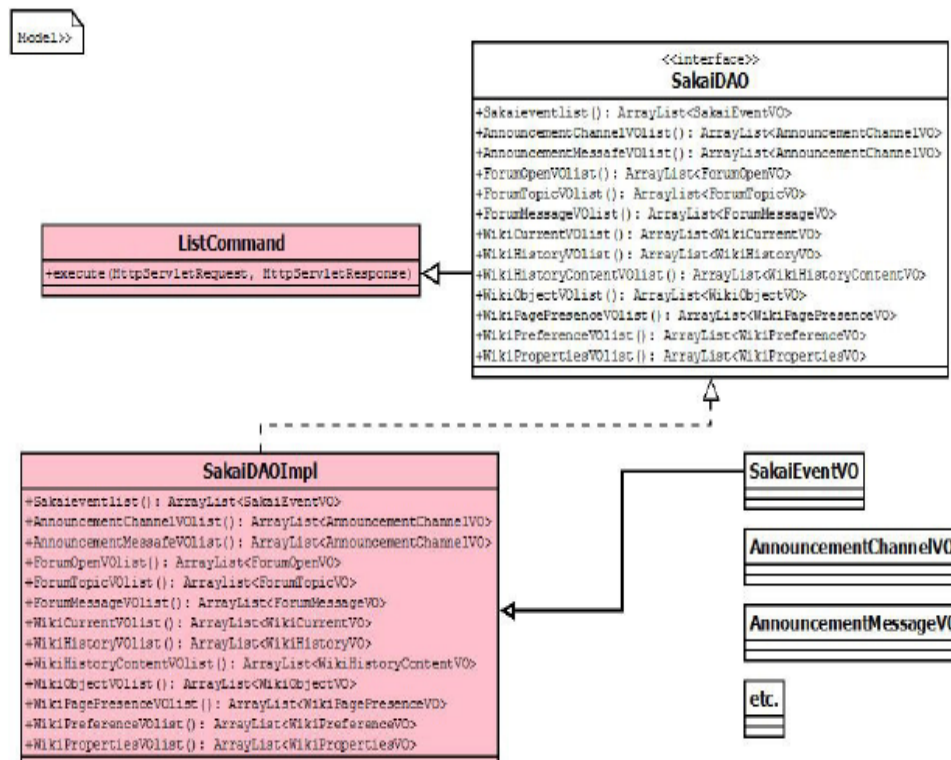
Modelo de integración para contratos, métricas y modelo DEVS.

**Paso 2: Acceso a la base de datos** Si bien este paso es transparente para el usuario final los diseñadores e implementadores deben cumplir con ciertos requisitos que se ajustan a las necesidades del modelo de integración aquí propuesto. En este sentido se establecen diferentes tipos de conexiones entre subsistemas a través de mensajes y secuenciación de ejecución de tareas (ej, relación toma resultado (5), sección 3). Por otro lado, es necesario implementar acciones de penetración entre sistemas que deben respetar una infraestructura tecnológica y de diseño, este es el caso de la vinculación que se propone en el modelo de integración entre el módulo Integrador y la base de datos que contiene las interacciones (relación acceso (2), sección 3).

En este caso, proponemos un modelo de acceso a base de datos a través de iBATIS teniendo en cuenta que la capa de abstracción será la interfaz con la capa de la lógica de la función transferencia, haciendo las veces de “facade” entre la aplicación y la persistencia. Teniendo en cuenta el patrón Data Access Object (DAO), nuestra implementación utilizó del framework DAO (ibatis-dao.jar). La capa de Framework de Persistencia será la interfaz con el gestor de Base de Datos ocupándose de la gestión de los datos mediante el framework SQL-MAP (ibatis-sqlmap.jar).

## 8.2. SEPI: una herramienta para el Seguimiento y Evaluación de Procesos Interactivos del DHD

En la siguiente Figura 3, mostraremos un ejemplo de las clases y relaciones que implementa el acceso a la tabla sakaiEvent de la base de datos Sakai.



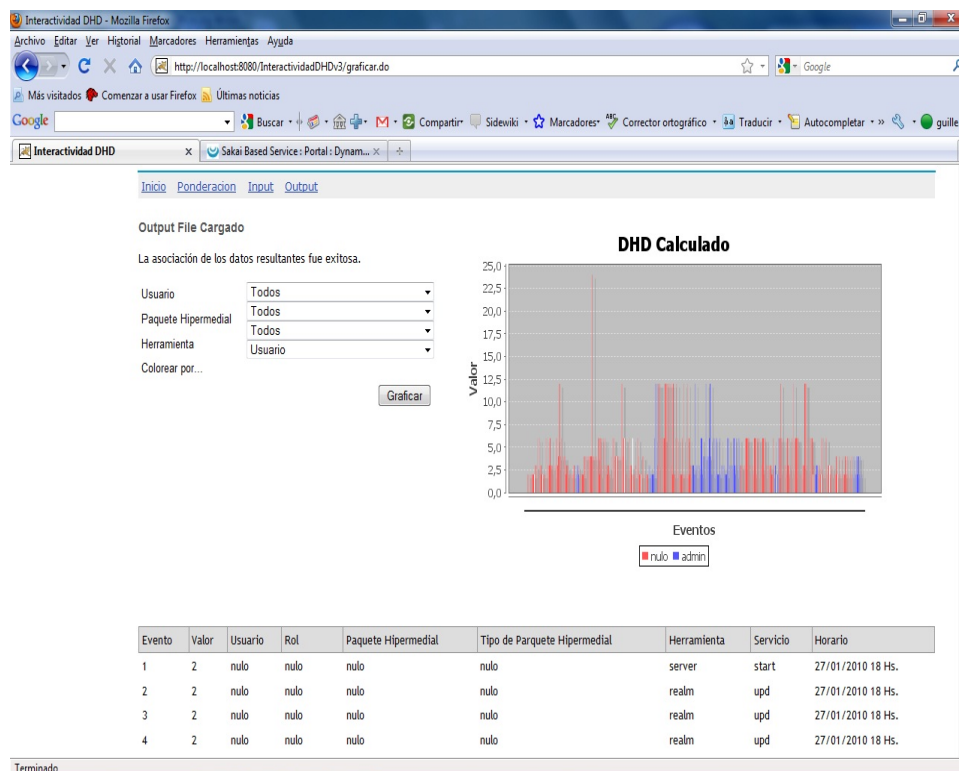
Estructura para el acceso a la Base de Datos.

**Paso 3: Configuración de coeficientes de la métrica** A partir del acceso a la base de datos, la aplicación genera un archivo `input.csv` en el cual se guardan los vectores (de ocho componentes) [12] que serán los datos de entrada del entorno utilizado para correr el modelo DEVS que integra las métricas (relación acceso (3), sección 3). Cada métrica directa tiene asociado un método de medición claramente especificado.

**Paso 4: Ejecución de la simulación** Una vez generados los valores de entrada correctos, estos serán utilizados en la simulación. Se ejecutará el modelo DEVS general como un módulo `model.exe` que posee dos parámetros, el archivo de entrada antes generado: `input.csv`, y el número de eventos totales (relación acceso (4), sección 3). Luego de la ejecución se genera el archivo `output.csv` el cual contiene los valores de niveles de interactividad para cada participación.

## 8.2. SEPI: una herramienta para el Seguimiento y Evaluación de Procesos Interactivos del DHD

**Paso 5: Lectura de resultados** A continuación, la Figura 4 muestra gráficamente a modo de ejemplo, los resultados obtenidos de Nivel de Interactividad para cada participación a través del tiempo en el entorno colaborativo de investigación del Programa DHD, antes mencionado. La interfase posibilita diversos filtrados y colorea los niveles de interactividad según los requerimientos del usuario (relación acceso (5), sección 3).



Estructura para el acceso a la Base de Datos.

**Paso 6: Insertar valores en los contratos** El resultado de este análisis es considerado una información de contexto, resignificando una característica del comportamiento de los participantes y atendiendo a la posibilidad de usar la información de los niveles de interactividad como parámetro context-aware de los contratos (relación acceso (6), sección 3).

### 8.2.5. Conclusiones

En este trabajo hemos presentado la herramienta integrada SEPI para el seguimiento y evaluación de procesos de interacción del DHD adaptado al entorno

## **8.2. SEPI: una herramienta para el Seguimiento y Evaluación de Procesos Interactivos del DHD**

---

colaborativo SAKAI. A su vez, se extendieron las propiedades expresivas de las reglas de coordinación de contratos del DHD a partir de los resultados de un mecanismo externo. De esta manera, es posible establecer un lazo de retroalimentación entre las acciones efectuadas en los entornos colaborativos y las que devengan de los contratos.

Además de las cualidades funcionales logradas, la herramienta SEPI concreta la integración de los subsistemas por medio de mensajes, ejecución de tareas y penetración, atendiendo a las líneas de diseño e implementación establecidas dentro del marco teórico-metodológico del DHD. La prospectiva actual se centra en la optimización de la herramienta, su prueba en diversos escenarios de uso y la puesta a disposición de la misma en los foros de discusión con la documentación correspondiente.

---





---

## Conclusiones y trabajos Futuros

---

### 9.1. Conclusiones

### 9.2. Lineas de trabajo futuras

Entre las conclusiones expuestas en la mencionada tesis, se argumenta sobre el impacto de los ContratosDHD, como pieza de software de primera clase, para la adaptación de los servicios de las herramientas dentro del framework colaborativo Sakai ([www.sakaiproject.org](http://www.sakaiproject.org)) desarrollándose en el corpus tanto aspectos relativos a la arquitectura como a su funcionalidad. Los aportes se centran por un lado en el desarrollo y análisis de las arquitecturas que implementan los ContratosDHD y el rol que esta componente cumple y, por otro lado, en el estudio y desarrollo de los aspectos funcionales de los ContratosDHD para su implementación teniendo en cuenta los requerimientos funcionales y, la forma de implementación en base a la tecnología y su descripción. En la prospectiva de trabajo planteada en la tesis, se desprende la necesidad de formalizar varios de los aspectos funcionales de los ContratosDHD con el propósito de generar casos de prueba que aseguren el correcto funcionamiento y además permitan establecer una original, eficiente y eficaz forma de corroborar aquellos requisitos y requerimientos que provienen de prácticas mediatizadas a través de redes sociotécnicas que tienen por objeto desarrollar procesos de formación superior, investigación o vinculación tecnológica.

## 9.2. Líneas de trabajo futuras

---

Consideramos que el logro de este propósito colaborará en la construcción efectiva del campo interdisciplinario que solicita la perspectiva DHD.

Como antecedente en la generación de casos de pruebas a través de testing basado en especificaciones “Z” se cuenta con el recorrido teórico y práctico del grupo de Ingeniería de Software del CIFASIS dirigido por el Mgs: Maximiliano Cristiá (co-director de este plan de trabajo) quien a su vez lidera como Profesor Titular las actividades de investigación y docencia de la cátedra de Ingeniería de Software de la Licenciatura en Ciencias de la Computación (UNR) donde el postulante se desempeña como integrante. Entre los resultados arribados, se cuenta con la producción Fastest, la cual es la primera implementación [ref registro propiedad] del Test Template Framework (TTF)<sup>1</sup>. El TTF es un método específico de testing basado en modelos. Como lo indica el TTF, Fastest recibe una especificación “Z” y genera de forma casi automática casos de prueba derivados de esa especificación<sup>2</sup>. La herramienta usa el framework CZT (<http://czt.sourceforge.net>).

En el marco de las tesis doctorales que se desarrollan en el Programa de I+D+T Dispositivos Hipermediales Dinámicos se han desarrollado trabajo interrelacionados que fundamentan el presente plan. En directa relación con la tesis de A. Sartorio, cabe mencionar la tesis doctoral del Ing. Guillermo Rodríguez “La teoría de los sistemas complejos aplicada al modelado de Dispositivos Hipermediales Dinámicos” (Dir. Dra. Patricia San Martín; Codir. Dr. Juan Carlos Gómez-CIFASIS: CONICET-UNR-UPCAM), donde se expone avances en el diseño, programación, implementación y vinculación con los ContratosDHD de una primera herramienta experimental integrada que implicó un desarrollo de software original a la que se la denominó “SEPI-DHD” (Seguimiento y Evaluación de los Procesos Interactivos del DHD). Sobre la aplicación de testing basado en modelos, en particular con Fastest a los ContratosDHD se ha abordado introductoramente una serie de problemas vinculados al testing de integración y a la relación entre comportamiento, estructura y verificación guiada por la arquitectura que posibilitan el estudio de la problemática. Seguidamente se citan referencias a los principales trabajos en los que se centra el marco general de la I+D que fundamentaron en la tesis doctoral sobre los contratos DHD y que se profundizarán en el presente plan de trabajo.

- Sartorio A. (2010). En el capítulo 4 de la tesis doctoral del postulante se describen los aspectos arquitectónicos de los DHD desde la conjunción de las siguientes tres perspectivas: Ingeniería Web, Arquitectura Web y la Ingeniería basada en modelos. Describiendo las propiedades dinámicas que aportan los ContratosDHD a través de los lenguajes Darwin y Wright Dinámico.
- Sartorio A. y Cristiá M. (2009). Presentaron una primera aproximación al

diseño e implementación de los ContratosDHD en un framework colaborativo Web.

- San Martín et al. (2008-2010) presentan el concepto de Dispositivo Hipermedial Dinámico desde un enfoque teórico, metodológico y tecnológico para la integración efectiva de las TIC y herramientas de la WEB 2.0, en contextos físico-virtuales educativos, investigativos y de producción en la actual Sociedad de la información.
- El proyecto SAKAI ([www.sakaiproject.org](http://www.sakaiproject.org)) desarrollado inicialmente por el MIT, aporta los programas fuentes de la aplicación en desarrollo. Su comunidad está formada por prestigiosas universidades y afiliaciones comerciales. Posee los espacios para cursos y para proyectos y está en producción en el Campus Virtual UNR (versión 2.7, <http://200.3.120.183>), el área Transdepartamental de Crítica de Arte del IUNA (versión 2.6, [www.mesadearena.edu.ar](http://www.mesadearena.edu.ar)) y el Programa Dispositivos Hipermediales Dinámicos donde se están efectuando experimentaciones con la versión 3.0. ([control.cifasis-conicet.gov.ar:8080](http://control.cifasis-conicet.gov.ar:8080))
- Kofman et al. (2008) desarrollaron un entorno como software libre en el CIFASIS (CONICET-UNR-UPCAM) para la simulación de sistemas dinámicos, incluyendo modelos DEVS.
- Sartorio, A., San Martín, P. Rodríguez, G. (2010) desarrollan un software con titularidad CONICET, actualmente en evaluación para su copyright en CESSI, que consta de los siguientes componentes: voluntarias del Programa KADO (Corea del Sur) para el Proyecto PIP N 0718 "Obra Abierta: DHD para educar e investigar" (CIFASIS).
- Rodríguez et al. (2010) desarrollaron SEPI-DHD como un primer prototipo de herramienta de código abierto para el Seguimiento y Evaluación de Procesos de Interacción del DHD. (Tesis doctoral del postulante, Cap. VI), en el marco de su pasantía en CIFASIS como becario de la ANPCyT y miembro del proyecto I+D "Obra Abierta" ya mencionado. Los componentes tecnológicos que complementan el desarrollo cumplen los estándares de los frameworks, en este caso se utilizan Servlets y Beans teniendo en cuenta el acceso a los servicios base.
  1. Una herramienta para la inyección de las propiedades de coordinación de contratos sensibles al contexto y dinámicos a los servicios del framework Sakai;
  2. Un framework Sakai modelo, con propiedades de coordinación de contratos sensibles al contexto y dinámicos.
  3. Una herramienta Sakai para la transformación de datos registrados por las interacciones producidas por los participantes del DHD. Se basa en la

## 9.2. Líneas de trabajo futuras

---

aplicación de una función transferencia orientada a la aplicación de métricas insertas en un modelo sistémico complejo construido con un formalismo de eventos discretos DEVS (desarrollada con la colaboración de Kibb Lee y Hyunah Woo,

### 9.2.1. Propuesta de actividades

Se continuará con la metodología de trabajo que se fundamenta en la Ingeniería de Software para la especificación formal de requerimientos, su transformación formal, tareas de integración-pruebas y validación automática; especializada en la pieza de software ContratosDHD.

El proceso de I+D se ha diseñado de manera flexible en tres fases. Cada una de ellas se constituye a su vez, en momentos dentro del proceso de desarrollo, pudiéndose superponer, ampliar y modificar. El tiempo de las fases se encuadra en los dos años propuestos. Algunas de estas fases se incluyen dentro de la metodología general planteada en el Proyecto “Técnicas de Ingeniería de Software aplicadas al Dispositivo Hipermedial Dinámico”.

De esta manera se continuará trabajando a partir de los casos de usos presentado en la tesis, pretendiendo ampliarlos y reformularlos para una descripción acorde a la vinculación con los casos de pruebas abstractos obtenidos. Las actividades de desarrollo de software continuarán con la mismas metodologías implementadas en el desarrollo de Fastest y la herramienta para la inyección de contratos sensibles al contexto en los DHD.

Se llevarán a cabo análisis experimental de la herramienta mediante la prueba con requerimientos lo más reales posible para luego comparar los resultados obtenidos con respecto a las metodologías tradicionales.

### 9.2.2. Factibilidad

Actualmente se cuenta con un prototipo avanzado de Fastest para ser utilizado por usuarios finales, con su correspondiente documentación del diseño del software y manuales de usuarios y publicaciones con referato que muestran experiencias de uso. En este plan de trabajo no se requiere infraestructura tecnológica específica. Se cuenta con el acceso a la bibliografía y recursos adecuados para las distintas fases de la ejecución en el CIFASIS.

El CIFASIS, como parte del CCT-Rosario y centro binacional, cuenta con un centro de cómputo donde se aloja uno de los servidores de desarrollo y producción del grupo DHD. Su puesta en marcha ya realizada y actual manutención se encuentra bajo la labor conjunta del personal del CIFASIS y de miembros del Programa DHD. Se cuenta también con un servidor del grupo DHD alojado y administrado en la sede de gobierno de la UNR y entrada remota a los servidores del Campus Virtual UNR, con la autorización de la Secretaría de Tecnologías Educativas y de Gestión de la UNR. La disponibilidad de fondos, no se constituye en una situación problemática para el desarrollo eficiente de lo planteado ya que la tecnología de hardware, insumos, bibliografía, etc. necesaria fue adquirida en el marco de distintos proyectos de I+D bajo titularidad de la Dra. Patricia San Martín, consolidando la viabilidad de las implementaciones.



---

# Bibliografía

---

- [1] Vicerrectoría de Investigaciones de la Universidad del Cauca. Informe Final del Proyecto RedPacíficoCyT. Popayán, Enero de 2002.
- [2] Jonathan Grudin. CSCW: History and Focus [en línea]. University of California. IEEE Computer, 27, 5, 19-26. 1994 [citado 2003-02-22]. Disponible en la Web: <http://www.ics.uci.edu/~grudin/Papers/IEEE94/IEEEComplastsub.html>
- [3] Jonathan Grudin. Groupware and social dynamics: eight challenges for developers [en línea]. University of California. Communications of the ACM, 37, 1, 92-105. 1994 [citado 2003-02-22]. Disponible en la Web: <http://www.ics.uci.edu/~grudin/Papers/CACM94/cacm94.html>
- [4] HAYA, P. A., ALAMÁN, X., AND MONTORO, G. A comparative study of communication infrastructures for the implementation of ubiquitous computing. UPGRADE, The European Journal for the Informatics Professional 2, 5 (2001).
- [5] NORMAN, D. The Invisible Computer. MIT Press, 1998.
- [6] HSU, Jeffrey. Collaborative Computing: Byte Magazin. Dic. 1.988
- [7] HERNANDEZ, Marcela y RODRIGUEZ Marcela. Trabajo en grupo y “groupware”. Bogota CIFIUNIANDES, 1.996, p 6.
- [8] SCHNAIDT, Patricia. Workflow Applications. An emerging Method for Sharing Work: LAN Magazine. Feb. 1992
- [9] HERNANDEZ, Marcela y RODRIGUEZ Marcela. Trabajo en grupo y “groupware”. Bogota . CIFIUNIANDES, 1.996, p 6.



- [10] Distante D., Tilley S. and Huang S. (2004b). Documenting software systems with views IV: documenting web transaction design with UWAT+. *Proceedings of the 22nd International Conference on Design of Communication (SIGDOC 2004)*, Memphis, TN, New York, NY: ACM Press, 10–13 October.
- [11] Gelernter D. and Carriero N. Coordination Languages and their Significance. *Communications ACM* 35, 2, pp. 97-107, 1992.
- [12] Andrade L. and Fiadeiro J.L. Interconnecting Objects via Contracts. In *UML'99 – Beyond the Standard*, R.France and B.Rumpe (eds), LNCS 1723, Springer Verlag 1999, 566-583.
- [13] Anderson M., Ball M., Boley H., Greene S., Howse N., Lemire D., McGrath S. (2003), RACOFI: A Rule-Appling Collaborative Filtering System. Pages 53–72 of Proc. COLA'03. IEEE/WIC. Halifax, Canada.
- [14] GONZÁLEZ, J. L., RUBIO, A., AND MOLL, F. Human powered piezoelectric batteries to supply power to wearable electronic devices. *International Journal of the Society of Materials Engineering for Resources* 10,1 (2002), 33-40.
- [15] ABOWD, G. D., ATKESON, C. G., HONG, J. L, LONG, S., KOOPER, R., AND PINKERTON, M. Cyberguide: A mobile context-aware tour guide. *Wireless Networks* 3, 5 (October 1997), 421-433.
- [16] AIRE, <http://aire.csail.mit.edu/>, 2003.
- [17] ABOWD, G. D. Classroom 2000: an experiment with the instrumentation of a living educational environment. *IBM System Journal* 38, 4 (1999), 508-530.
- [18] ABASCAL, J., CAGIGAS, D., GARAY, N., AND GARDEAZABAL, L. MOBILE interface for a smart wheelchair. In *Fourth International Symposium on Human Computer Interaction with Mobile Devices* (Pisa (Italy), 2002), F. Paterno, Ed., vol. 411 of LNCS, Springer-Verlag, pp. 373-377.
- [19] CLARK, H. H. Using language. Cambridge University Press, 1996.
- [20] OLIVER, N. Towards Perceptual Intelligence: Statistical Modeling of Human Individual and Interactive Behaviors. PhD thesis, MIT Media Lab, 2000.
- [21] PATERNO, F., AND SANTORO, C. One model, many interfaces. In *Computer-Aided Design of User Interfaces III*. (Dordrecht, Hardbound, May 2002), C. Kolski and J. Vanderdonckt, Eds., GADUI, Kluwer Academic Publishers, pp. 143-154.
- [22] ALI, M. F., PÉREZ-QUIÑONES, M. A., ABRAMS, M., AND SHELL, E. Building Multi-Platform User Interfaces with UIML, computer-aided design of user interfaces iii (cadui) ed. Kluwer Academic Publishers, Dordrecht, Hardbound, 2002, ch. 22, pp. 225-236.

- [23] PUERTA, A., AND EISENSTEIN, J. XIML: A universal language for user interfaces. White Paper, 2001. <http://www.ximl.org/Docs.asp>.
- [24] ALONSO, N., BALAGUERA, A., JUNYENT, E., LAFUENTE, A., LÓPEZ, J. B., LORES, J., MUÑOZ, D., PÉREZ, M., AND TARTERA, E. Virtual reality as an extension of the archaeological record: The reconstruction of the iron age fortress of Vilars. In CAÁ, Computer Applications, and quantitative methods in Archaeology (Ljubiana, Eslovenia, 2000).
- [25] UWA (2001), UWA Requirements Elicitation: Model, Notation, and Tool Architecture
- [26] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed Requirements Acquisition", Science of Computer Programming; Vol. 20, 1993
- [27] Garzoto F., Schwabe D. and Paolini P. (1993) HDM-A Model Based Approach to Hypermedia Application Design. ACM Transactions on Information Systems, 11 (1), pp 1-26.
- [28] Baeza, Y. R. y Ribeiro, N. B. (1999) Modern Information Retrieval, Nueva York, Addison Wesley, ACM, disponible en: <http://www.dcc.ufmg.br/irbook/print/chap10.ps.gz>
- [29] Balabanovic, M. y Shoham, Y. (1997) Fab: Content-based, collaborative recommendation, Nueva York, Communications of the ACM, disponible en: <http://citeseer.ist.psu.edu/balabanovic97combining.html>
- [30] Bradshaw, S., and Hammond, K. J. (1999) Mining Citation Text as Indices for Document, Medford Nueva York, en Proceedings of the ASIS 1999 Annual Conference.
- [31] Brown, P. J. y Jones, G. J. F. (2002) Exploiting contextual change in context-aware retrieval, Madrid, ACM Press, New York, Proceedings of the 17th ACM Symposium on Applied Computing (SAC 2002), disponible en <http://portal.acm.org/citation.cfm?id=508917coll=GUIDEId=GUIDEcfid=16172620CFTOKEN=19042730>
- [32] Berners-Lee, Tim. (1998), A roadmap to the Semantic Web. disponible en: <http://www.w3.org/DesignIssues/Semantic.html>.
- [33] Brown, P. J., Jones, G.J.F., (2000) Context-aware retrieval: exploring a new environment for information retrieval and information filtering. Personal and Ubiquitous Computing, 5(4):253–263,
- [34] Brown, P., Burleston, W., Lamming, M., Rahlff, O., Romano, G., Scholtz, J., and Budzik, J. y Hammond, K. (2000) User Interactions

- with Everyday Applications as Context for Just-in-time Information Access. Proceedings of Intelligent User Interfaces 2000. ACM, disponible en: <http://portal.acm.org/citation.cfm?id=325776coll=GUIDEdl=GUIDECFID=16172620CFTOKEN=19042730>
- [35] Cabero J. (2006) Bases pedagógicas del e-learning, Sevilla, Revista de Universidad y Sociedad del Conocimiento Vol. 3 - N.º 1, disponible en: <http://www.raco.cat/index.php/RUSC/article/viewFile/49343/50232>
- [36] Calvino, I. (2001), Colección de Arena, España, Siruela. Cheng, I. and Wilensky, R. (1997) An Experiment in Enhancing Information Access by Natural Language. Technical Report. California, University of California at Berkeley, disponible en: <http://www.eecs.berkeley.edu/Pubs/TechRpts/1997/CSD-97-963.pdf>
- [37] De Kerckhove, D. (1999), La piel de la cultura. Investigando la nueva realidad electrónica, España, Gedisa.
- [38] Dey, A. K. y Abowd, G. D. (2000) Towards a Better Understanding of Context and Context-Awareness. Presented at the CHI 2000 Workshop on The What, Who, Where, When, Why and How of Context-Awareness, Georgia Institute of Technology.
- [39] Dourish, P. (2004) What we talk about when we talk about context, Roma, Personal and Ubiquitous Computing, vol. 8, no. 1, 2004, pp. 19–30, disponible en: <http://www.springerlink.com/content/y8h8l9me8yabycl3/>
- [40] Eco, U. (1998) Los límites de la interpretación, Barcelona, Lumen. Elliott, G. T. and B. Tomlinson (2006), Personalsoundtrack: context-aware playlists that adapt to user pace. En: CHI '06: CHI '06 extended abstracts on Human factors in computing systems. ACM Press. New York, NY, USA. pp. 736–741.
- [41] Fu, X., Budzik, J. y Hammond, K. (2000) Mining Navigation History for Recommendation. Proceedings of Intelligent User Interfaces ACM, disponible en: <http://infolab.northwestern.edu/infolab/downloads/papers/paper10081.pdf>  
Glover, E. J., Lawrence, S., Gordon, M. D., Birmingham, W. P., y Giles, C. L. (2000) Web search – your way. Communications of the ACM, disponible en: <http://citeseer.ist.psu.edu/glover00web.html>
- [42] James R., Jacobson I., Booch Grady (2003) “UML 2.0” The Unified Modeling Language Reference Manual, Londres, Addison-Wesley Object Technology Series, disponible en: <http://portal.acm.org/citation.cfm?coll=GUIDEdl=GUIDEid=294049>
- [43] Gross, T., Braun, S., Krause, S.(2006) MatchBase: A Development Suite for Efficient Context-Aware Communication, Los Alamitos, Estados

- Unidos, Proceedings. PDP '06. IEEE Computer Society, disponible en: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=1613289](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1613289)
- [44] Jay, M. (2003), Campos de fuerza. Entre la historia intelectual y la crítica cultural, Buenos Aires, Paidós.
- [45] Jones, G.J.F. and Brown, P.J., (2004) Context-aware retrieval for ubiquitous computing environments, Invited paper in Mobile and ubiquitous information access, Springer Lecture Notes in Computer Science, disponible en: <https://www.springerlink.com/content/b33d471j4xa72tqq/resourcesecured/?target=fulltext.pdf>
- [46] Kruger, A.; Giles, C.L.; Coetzee, F.; Glover, E.; Flake, G.; Lawrence, S. and Omlin, C. (2000) DEADLINER: Building a new niche search engine. Virginia, Estados Unidos, In Ninth International Conference on Information and Knowledge Management, CIKM, disponible en: <http://portal.acm.org/citation.cfm?id=354756.354829>
- [47] Lawrence, S. (2000) Context in Web Search. IEEE Data Engineering Bulletin, disponible en: <http://www.fravia.com/library/context-deb00.pdf>
- [48] Lawrence, S.; Giles, C.L.; Bollacker, K. (1999) Digital Libraries and Autonomous
- [49] Citation Indexing, IEEE Computer, disponible en: <http://doi.ieeecomputersociety.org/10.1109/2.769447>
- [50] Lieberman, H. (1995) Letizia: An agent that assists web browsing. Proceedings 14th International Conference Artificial intelligence (IJCAI), 924-929, disponible en: <http://web.media.mit.edu/~lieber/Lieberary/Letizia/Letizia-AAAI/Letizia.html>
- [51] Rhodes, B. (2000) Just in Time Information Retrieval. PhD thesis. Massachusetts Institute of Technology, disponible en: <http://www.research.ibm.com/journal/sj/393/part2/rhodes.html>
- [52] Rodríguez, M., y Preciado, A. (2004) An Agent Based System for the Contextual Retrieval of Medical Information., In AWIC 2004, LNAI 3034, pp. 64-73, disponible en: <http://www.springerlink.com/content/8apnh5evx1n7tylj/>
- [53] Salton, G. y Buckley, C. (1990) Improving Retrieval Performance by Relevance Feedback. Journal of the American Society for Information Science, disponible en: <http://www.scils.rutgers.edu/~muresan/IR/Docs/Articles/jasistSalton1990.pdf>
- [54] San Martín P., Sartorio A. (2006), Implementaciones de entornos e-learning en la formación de arquitectos. Hacia una aplicación contex-aware dinámica física-digital en Rodríguez Barros, Diana. (Comp.) Experiencia Digital. Usos, prácticas y estrategias en talleres de arquitectura y diseño en entornos virtuales. Mar del Plata, Universidad de Mar del Plata, 2006, pp. 195-204.

- [55] San Martín P., Sartorio A., Rodríguez G. (2006) Una mesa de arena para Investigar y Aprender en Contextos físicos-virtuales-interactivos-comunicacionales de Educación Superior. Actas del XV Encuentro Internacional de Educación a distancia. UDG. Guadalajara, México.
- [56] Schmidt, A. (2005) Bridging the Gap Between E-Learning and Knowledge Management with Context-Aware Corporate Learning Solutions. Proceedings WM '05, Springer LNCS, 3782, disponible en: [http://publications.professionallearning.eu/Schmidt\\_LOKMOL05\\_Extended.pdf](http://publications.professionallearning.eu/Schmidt_LOKMOL05_Extended.pdf)
- [57] San Martín, P., Sartorio, A., Guarnieri, G., Rodríguez, G.: Hacia un dispositivo hipermedial dinámico. Educación e Investigación para el campo audiovisual interactivo. Universidad Nacional de Quilmes (UNQ). ISBN: 978-987-558-134-0. (2008)
- [58] Sartorio, A., San Martín, P.: Sistemas Context-Aware en dispositivos hipermediales dinámicos para educación e investigación. En San Martín P., Sartorio A., Guarnieri G., Rodríguez G. Hacia un dispositivo hipermedial dinámico. Educación e Investigación para el campo audiovisual interactivo. Universidad Nacional de Quilmes (UNQ). ISBN: 978-987-558-134-0. (2008)
- [59] Sartorio, A.: Un modelo comprensivo para el diseño de procesos en una Aplicación E-Learning. XIII Congreso Argentino de Ciencias de la Computación. CACIC 2007. ISBN 978-950-656-109-3
- [60] Sartorio, A.: Un comprensivo modelo de diseño para la integración de procesos de aprendizajes e investigación en una aplicación e-learning. Edutec2007. Inclusión Digital en la Educación Superior Desafíos y oportunidades en la Sociedad de la Información. ISBN 978-950-42-0088-8. (2007)
- [61] Sartorio A. Los contratos context-aware en aplicaciones para educación e investigación. En San Martín, P., Sartorio, A., Guarnieri, G., Rodríguez, G.: Hacia un dispositivo hipermedial dinámico. Educación e Investigación para el campo audiovisual interactivo. Universidad Nacional de Quilmes (UNQ). ISBN: 978-987-558-134-0. (2008)
- [62] Hartmann J., Huang S., and Tilley S. Documenting Software Systems with Views II: An Integrated Approach Based on XML. Proceedings of the 19th Annual International Conference on Systems Documentation (SIGDOC 2001: October 21-24, 2001; Santa Fe, NM), pp. 237-246. ACM Press: New York, NY, 2001.
- [63] Tilley S. and Huang S. Documenting Software Systems with Views III: Towards a Task-Oriented Classification of Program Visualization Techniques. Proceedings of the 20th Annual International Conference on Systems Documentation (SIGDOC 2002: October 20-23, 2002; Toronto, Canada), pp. 226-233. ACM Press: New York, NY, 2002.

- [64] Tilley S., Müller H. and Orgun M. Documenting Software Systems with Views. *Proceedings of the 10th Annual International Conference on Systems Documentation (SIGDOC '92: October 13-16, 1992; Ottawa, Canada)*, pp. 211-219. ACM Press: New York, NY, 1992.
- [65] UWA Consortium.: Ubiquitous web applications. Proceedings of The eBusiness and eWork Conference (e2002), 16–18 October, Prague, Czech Republic.(2002)
- [66] Dey, A.K., Salber, D., Abowd, G.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, anchor article of a special issue on Context-Aware Computing. *Human-Computer Interaction (HCI) Journal*, Vol. 16 (2-4), pp. 97-166. (2001)
- [67] Brambilla, M., Ceri, S., Fraternali, P., Manolescu I.: Process modeling in web applications. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, in print.
- [68] Andrade, L., Fiadeiro, J.L.: Interconnecting Objects via Contracts. In *UML'99 – Beyond the Standard*, R.France and B.Rumpe (eds), LNCS 1723, Springer Verlag (1999)
- [69] Obra Abierta: Proyecto de I&D (CONICET-CIFASIS), que se centra en el desarrollo e implementación de Dispositivos Hipermediales context-aware Dinámico para investigar y aprender en contextos físicosvirtuales de educación superior. Directora: Patricia San Martín
- [70] Meyer, B.: Applying Design by Contract, *IEEE Computer*, 40-51. (1992)
- [71] Proyecto de investigación CIFASIS-UNR llamado: "Técnicas de Ingeniería de software aplicadas al Dispositivo hipermedial dinámico. Director: Mg. Maximiliano Cristiá". Dicha propuesta consiste en la incorporación a Sakai de un framework para la coordinación de contratos (68; 78)
- [72] The Oblog Corporation. The Oblog Specification Language, disponible en: "[http :  
//www.oblog.com/tech/spec.html](http://www.oblog.com/tech/spec.html)"
- [73] Dowling J, Cahill, V.: Dynamic software evolution and the k-component model. In: *Proc. of the Workshop on Software Evolution, OOPSLA*. (2001)
- [74] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: *Pattern-Oriented Software Architecture*. John Wiley. (1996)
- [75] Gelernter, D., Carriero, N.: Coordination Languages and their Significance. *Communications ACM* 35, 2, pp. 97-107, (1992)
- [76] Sartorio, A., Guarnieri, G., San Martín, P.: Students' interaction in an e-learning contract context-aware application with associated metric", *Actas del*

- INTED2007, International Technology, Education and Development Conference, IATED, Valencia, España. (2007).
- [77] Gamma, E., Helm R., Johnson R., Vlissides, J.: Design Patterns: Elements of Reusable Object Oriented Software, Addison-Wesley (1995)
- [78] Davy, A., Jennings, B.: Coordinating Adaptation of Composite Services. Proceedings of the Second International Workshop on Coordination and Adaptation Techniques for Software Entities. WCAT'05 Glasgow , Scotland (2005)
- [79] Dourish, P.: What we talk about when we talk about context. Personal and Ubiquitous Computing, vol. 8, N° 1, Roma, 2004, pp. 19-30, disponible en <http://www.springerlink.com/content/y8h8l9me8yabycl3/>
- [80] Houben, G.: Adaptation Control in Adaptive Hypermedia Systems. en Adaptive Hypermedia Conference. AH2000. Trento, Italia. Agosto. LNCS, vol. 1892. Springer-Verlag, pp. 250-259. (2000)
- [81] Programa I+D+T “Dispositivos Hipermediales Dinámicos”, <http://www.mesadearena.edu.ar>
- [82] San Martin, P.; Sartorio, A.; Guarnieri, G.; Rodríguez, G.: Hacia la construcción de un dispositivo hipermedial dinámico. Educación e investigación para el campo audiovisual interactivo. Universidad Nacional de Quilmes Editorial, Buenos Aires (2008).
- [83] Dey, A.K., Salber, D., Abowd, G.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, anchor article of a special issue on Context-Aware Computing. Human-Computer Interaction (HCI) Journal, Vol. 16 (2-4), pp. 97-166. (2001).
- [84] Sartorio, A.; Cristiá, M.: Primera aproximación al diseño e implementación de los DHD. XXXIV Congreso Latinoamericano de Informática, CLEI, (2008).
- [85] Rivera, M.B.; Molina, H.; Olsina, L. Sistema Colaborativo de Revisión para el soporte de información de contexto en el marco C-INCAMI, XIII Congreso Argentino de Ciencias de la Computación, CACIC, (2007).
- [86] Gell-Mann, M.: El quark y el jaguar. Aventuras en lo simple y lo complejo. Tusquets, Barcelona (1995).
- [87] Zeigler, B.; King, Tan Gon; Praehofer, H.: Theory of modeling and Simulation. Second edition, Academic Press, New York (2000).
- [88] Zeigler, B.: Theory of modeling and Simulation. John Wiley Sons, New York (1976).

- [89] Olsina L., Martín M.: *Ontology for Software Metrics and Indicators*, Journal of Web Engineering, Rinton Press, US, Vol 2 N° 4, pp. 262-281, ISSN 1540-9589.
- [90] Olsina L., Molina H; Papa F.: *Organization-Oriented Measurement and Evaluation Framework for Software and Web Engineering Projects*, Lecture Notes in Computer Science of Springer, LNCS 3579, Intl Congress on Web Engineering, (ICWE05), Sydney, Australia, July 2005.
- [91] <http://sakai.bestgrid.org/portal/help/TOCDisplay/content.hlp?docId=armh#s-realms>
- [92] <http://personales.upv.es/darolmar/cursos/Manual%20Sakai.pdf>
- [93] Sheng, Q.Z. and Benatallah, B. (2005) 'ContextUML: a UML-based modeling language for model-driven development of context-aware web services', Proceedings of the International Conference on Mobile Business (ICMB'05), pp.206–212.
- [94] D. Salber, A. K. Dey, and G. D. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In Proc. of the Conference on Human Factors in Computing Systems (CHI'99), Pittsburgh, PA, USA, May 1999.
- [95] Olsina, L., Rossi, G. Measuring Web Application Quality with WebQEM, IEEE Multimedia, 9(4), 2002, pp. 20-29.
- [96] San Martín, P.; Guarnieri, G.; Rodríguez, G.; Bongiovani, P.; Sartorio, A. El Dispositivo Hipermedial Dinámico Campus Virtual UNR, Secretaría de Tecnologías Educativas y de Gestión, UNR, Rosario (2010). Disponible en: <http://rehip.unr.edu.ar/handle/2133/1390>.
- [97] Dujmovic J., Bazucan A., A Quantitative Method for Software Evaluation and its Application in Evaluating Windowed Environments, San Francisco, Estados Unidos, IASTED Software Engineering Conference, San Francisco, (1997).
- [98] Campus Virtual UNR, <http://www.campusvirtualunr.edu.ar>.
- [99] Foucault, M.: *Saber y verdad*. La Piqueta, Madrid (1991).
- [100] San Martín, P.; Guarnieri, G.; Rodríguez G.; Bongiovani, P.; Sartorio A. El dispositivo Hipermedial Dinámico Campus Virtual UNR. Secretaría de Tecnologías Educativas y Gestión. UNR, Rosario (2010) disponible en: <http://rehip.unr.edu.ar/handle/2133/1390>.
- [101] Dey, A.K.; Salber, D.; Abowd, G.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, anchor article of a special issue on Context-Aware Computing. Human-Computer Interaction (HCI) Journal, Vol. 16 (2-4), pp. 97-166. (2001).



- [102] Gell-Mann, M.: El quark y el jaguar. Aventuras en lo simple y lo complejo. Tusquets, Barcelona (1995).
- [103] García, R.: Sistemas Complejos. Conceptos, métodos y fundamentación epistemológica de la investigación interdisciplinaria, Gedisa, Buenos Aires (2007).
- [104] Programa I+D+T “Dispositivos Hipermediales Dinámicos”, <http://www.mesadearena.edu.ar>. Proyecto PIP N 718 (CONICET) “Obra Abierta: DHD para educar e investigar.” Dir. Dra. Patricia San Martín.
- [105] San Martín, P.; Sartorio, A.; Guarnieri, G.; Rodríguez, G.: Hacia la construcción de un dispositivo hipermedial dinámico. Educación e investigación para el campo audiovisual interactivo. Universidad Nacional de Quilmes Editorial, Buenos Aires (2008).
- [106] Zeigler, B.; King, Tan Gon; Praehofer, H.: Theory of modeling and Simulation. Second edition, Academic Press, New York. (2000).
- [107] Zeigler, B.: Theory of modeling and Simulation. John Wiley Sons, New York. (1976).
- [108] Rivera, M.B.; Molina, H.; Olsina, L.: Sistema Colaborativo de Revisión para el soporte de información de contexto en el marco C-INCAMI, XIII Congreso Argentino de Ciencias de la Computación, CACIC. (2007).
- [109] Sartorio, A.; Cristiá, M.: First Approximation to DHD Design and Implementation. Clei electronic journal, Vol.12 N. 1. (2009).
- [110] Meyer, B.: Applying Design by Contract, IEEE Computer, 40-51. (1992).
- [111] Rodríguez, G.; San Martín, P.; Sartorio, A.: Aproximación al modelado del componente conceptual básico del Dispositivo Hipermedial Dinámico. XV Congreso Argentino de Ciencias de la Computación. CACIC 2009. San Salvador de Jujuy. (2009).
- [112] Rodríguez, G.: Desarrollo e implementación de métricas para el análisis de las interacciones del Dispositivo Hipermedial Dinámico. Jornadas Argentinas de Informática. JAIIO 2010, Caba. (2010).
- [113] PowerDEVS 2.0 Integrated Tool for Edition and Simulation of Discrete Event Systems. Desarrollado por: Esteban Pagliero, Marcelo Lapadula, Federico Bergero. Dirigido por Ernesto Kofman. (<http://www.fceia.unr.edu.ar/lsd/powerdevs/index.html>).
- [114] Rumbaugh, J.; Jacobson, I.; Booch, G.: The Unified Modeling Language Reference Manual. Addison Wesley Logman, Inc.; Massachusetts (1999).

- [115] Sartorio, A., Guarnieri, G., San Martín, P.: Students' interaction in an e-learning contract context-aware application with associated metric", Actas del INTED2007, International Technology, Education and Development Conference, IATED, Valencia, España. (2007).
- [116] Paradkar, a. (2005) Case studies on fault detection e effectiveness of model based test generation techniques. In: WORKSHOP ON ADVANCES IN MODEL-BASED TESTING (A-MOST), 1., 2005, St. Louis, MO, USA. Proceedings... New York, NY, USA: ACM, 2005. p. 17. 8, 60
- [117] Cristiá, M (et. al.) (2009). Dirección Nacional del derecho de autor. Inscripción de Obra Publicada (software), expediente número 761794, Fastest 1.3.
- [118] Stocks, P. ; Carrington, D. (1996). A Framework for Specification-Based Testing, IEEE Trans. on Soft. Eng., vol. 22, no. 11, pp. 777–793, Nov. 1996.
- [119] Hierons, R. et.al. (2009). Using formal specifications to support testing. ACM Comput. Surv., vol. 41, no. 2, pp. 1–76, 2009.
- [120] Utting, M.; Legeard, B. (2006) Practical Model-Based Testing: A Tools Approach. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006
- [121] ISO, "Information Technology – Z Formal Specification Notation – Syntax, Type System and Semantics," International Organization for Standardization, Tech. Rep. ISO/IEC 13568, 2002.
- [122] Rémi Douence y Mario Sudholt. The Next 700 Reflective Object-Oriented Languages. Technical Report 99-1-INFO, Ecole des Mines de Nantes, 1999.
- [123] Jim Dowling, Tilman Schafer, Vinny Cahill, Peter Haraszti y Barry Redmond. Using Reflection to Support Dynamic Adaptation of System Software: A Case Study Driven Evaluation. En Walter Cazzola, Robert J. Stroud y Francesco Tisato, editores, Reflection and Software Engineering, volumen 1826 de Lecture Notes in Computer Science, págs. 171–190. Springer Verlag, Junio 2000.
- [124] Naranker Dulay. A Configuration Language for Distributed Programming. Tesis Doctoral, Imperial College of Science, Technology and Medicine. University of London, Febrero 1990.
- [125] Naranker Dulay. arwin Language Reference Manual. Technical report, Department of Computing, Imperial College, 1992.
- [126] Walter J. Ellis, Richard F. Hilliard III, Peter T. Poon, David Rayford, Thomas F. Saunders, Basil Sherlund y Ronald L. Wade. Toward a Recommended Practice for Architectural Description. En Proceedings of Second IEEE International Conference on Engineering of Complex Computer Systems, Montreal, Quebec, Canadá, Octubre 1996. IEEE Architecture Planning Group.

- [127] Uffe H. Engberg y Mogens Nielsen. A Calculus of Communicating Systems with Label Passing – Ten Years After. En Plotkin et al. [PST00].
- [128] Markus Endler. A Language for Implementing Generic Dynamic Reconfigurations of Distributed Programs. En Proceedings of the 12th Brazilian Symposium on Computer Networks, págs. 175–187, Curitiba, Mayo 1994.
- [129] Patrick Thomas Eugster. Type-Based Publish/Subscribe. Tesis Doctoral, Ecole Polytechnique Fédérale de Lausanne, Diciembre 2001.
- [130] Markus Endler y Jiawang Wei. Programming Generic Dynamic Reconfigurations for Distributed Applications. En Proceedings of the 1st International Workshop on Configurable Distributed Systems, págs. 68–79. IEE, 1992.
- [131] Svend Frølund y Gul Agha. A Language Framework for Multi-Object Coordination. En Nierstrasz [Nie93], págs. 346–360.
- [132] Svend Frølund y Gul A. Agha. Abstracting Interactions Based on Message Sets. En Object-Based Models and Languages for Concurrent Systems, volumen 924 de Lecture Notes in Computer Science, págs. 107–124. Springer-Verlag, 1996.
- [133] Jacques Ferber. Computational Reflection in Class Based Object Oriented Languages. En Meyrowitz [Mey89], págs. 317–326.
- [134] Nissim Francez y Ira R. Forman. Interacting Processes: A Multiparty Approach to Coordinated Distributed Programming. Addison-Wesley, 1996.327
- [135] Rost, A. “Pero, ¿De qué hablamos cuando hablamos de interactividad?”, Congresos ALAIC/IBERCOM 2004, La Plata, 2004.
- [136] Silva, M. “Educación Interactiva. Enseñanza y aprendizaje presencial y on-line”, Gedisa, Barcelona, 2005.
- [137] Templ J.: “Metaprogramming in Oberon”, ETH Dissertation No. 10655, Zurich, 1994
- [138] Leavens G.T, Baker A.L., Ruby C.: “JML: A Notation for Detailed Design”, In Haim Kilov, Bernhard Rumpe, and Ian Simmonds (editors), Behavioral Specifications of Businesses and Systems, chapter 12, pages 175-188. Copyright Kluwer, 1999
- [139] America, P. Designing an object-oriented programming language with behavioural subtyping. In Foundations of Object-Oriented Languages, REX School/-Workshop, Noordwijkerhout, The Netherlands, May/June 1990, Lecture Notes in Computer Science, pages 60–90. Springer-Verlag, 1991.
- [140] Findler R.B., Felleisen M.: “Contract Soundness for Object-Oriented Languages”, in Proceedings of OOPSLA 2001, ACM, 2001

- [141] Cicalese C.T.T., Rotenstreich S.: "Behavioral Specification of Distributed Software Component Interfaces", IEEE Computer, July 1999
- [142] Kramer R.: "iContract - The Java Design by Contract Tool", Proceedings of TOOLS USA '98 conference, IEEE Computer Society Press, 1998
- [143] Ensling O.: "iContract: Design by Contract in Java", JavaWorld, November 2001, see [http://www.javaworld.com/javaworld/jw-02-2001/jw-0216-cooltools\\_p.html](http://www.javaworld.com/javaworld/jw-02-2001/jw-0216-cooltools_p.html) (last visited: March 2002)
- [144] Parasoft: "Automatic Java Software and Component Testing: Using Jtest to Automate Unit Testing and Coding Standard Enforcement", see <http://www.parasoft.com/jsp/products/article.jsp?articleId=839&product=Jtest> (last visited: July 2002)
- [145] Parasoft: "Using Design by Contract to Automate Software and Component Testing", see <http://www.parasoft.com/jsp/>
- [146] Rogers P.: "J2SE 1.4 premieres Java's assertion capability" – Part 1, JavaWorld, November 2001, see [http://www.javaworld.com/javaworld/jw-11-2001/jw-1109-assert\\_p.html](http://www.javaworld.com/javaworld/jw-11-2001/jw-1109-assert_p.html) (last visited: March 2002)
- [147] Rogers P.: "J2SE 1.4 premieres Java's assertion capability" – Part 2, JavaWorld, November 2001, see [http://www.javaworld.com/javaworld/jw-12-2001/jw-1214-assert\\_p.html](http://www.javaworld.com/javaworld/jw-12-2001/jw-1214-assert_p.html) (last visited: March 2002)
- [148] Sun Microsystems: Java Assertion Facility - <http://java.sun>
- [149] Jakob E. Bardram. The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications. In Hans Gellersen, Roy Want, and Albrecht Schmidt, editors, Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005), volume 3468 of Lecture Notes in Computer Science, pages 98–115, Munich, Germany, May 2005. Springer Verlag.
- [150] Duncan A., Hölzle U.: "Adding contracts to Java with handshake", Technical Report TRCS98-32, The University of California at Santa Barbara, December 1998  
Karaorman96 Karaorman M., Hölzle U, Bruno J.: "jContractor: A reflective Java library to support design by contract", in Proceedings of Meta-Level Architectures and Reflection, Lecture Notes in Computer Science (LNCS), Volume 1616, Springer International, 1996
- [151] Bartečko D., Fischer C., Möller M., Wehrheim H.: „Jass – Java with Assertions“, Electronic Notes in Theoretical Computer Science, Proceedings of RV 01, Paris, France, Volume 55, Issue 2, July 2001

- [152] Dey, A.K., Salber, D., Abowd, G. "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, anchor article of a special issue on Context-Aware Computing", pp. 97-166, Human-Computer Interaction (HCI) Journal, Vol. 16 (2-4), 2001.
- [153] Brusilovsky, P. "Methods and techniques of adaptive hypermedia. User Modeling and User Adapted Interaction", Springer Netherlands Ed., Berlín, 1996.
- [154] Dey, A.K., Salber, D., Abowd, G. "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, anchor article of a special issue on Context-Aware Computing", pp. 97-166, Human-Computer Interaction (HCI) Journal, Vol. 16 (2-4), 2001.
- [155] McConnell, Steve (1996). Rapid Development: Taming Wild Software Schedules, 1st ed., Redmond, WA: Microsoft Press. ISBN 1-55615-900-5.
- [156] Wiegers, Karl E. (2003). Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle, 2nd ed., Redmond: Microsoft Press. ISBN 0-7356-1879-8.
- [157] Andrew Stellman and Jennifer Greene (2005). Applied Software Project Management. Cambridge, MA: O'Reilly Media. ISBN 0-596-00948-8.
- [158] IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications -Description