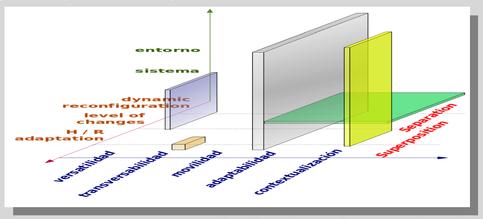
Los contratos sensibles al contexto para los Dispositivos Hipermediales Dinámicos

Directora: Dra. Patricia San Martín - Becarios: A. Sartorio, G. Rodriguez y G. Guarnieri
www.mesadearena.edu.ar

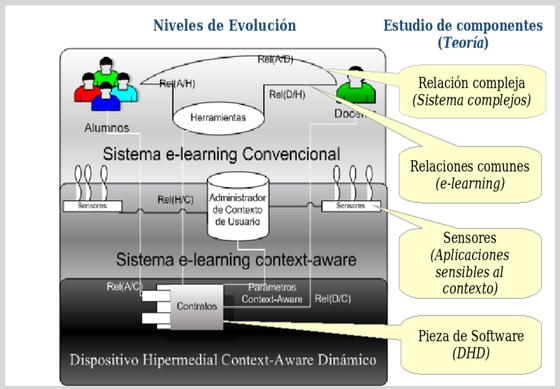
El DHD y sus Requerimientos

Definición: Un **DHD** no es solamente la tecnología, sino fundamentalmente la trama entre los individuos y ella, es decir los vínculos que se generan entre los sujetos intervinientes, la tecnología soporte, los contenidos disciplinares involucrados, la organización y la institución, y donde podemos observar la calidad de los procedimientos en el marco de una ética de participación responsable (San Martín et al, 2008).



Técnicamente un DHD es un diseño de software donde se combinan **Estilos Arquitectónicos** y **Patrones de Diseños** para conformar un modelo de integración de **componentes autónomas** y **de conección** que resuelven partes de los requerimientos enunciados en la definición de los DHD [clei2009].

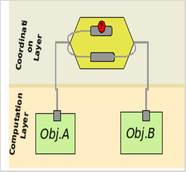
Modelo Evolutivo

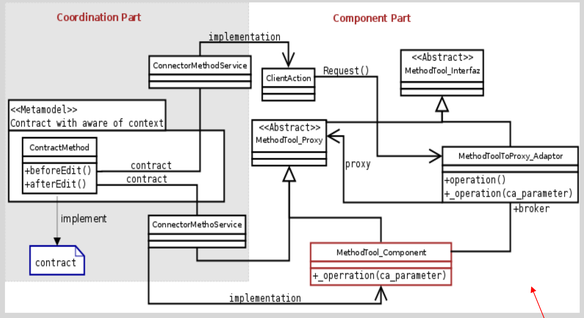


Coordinación de Contratos

El contrato es una abstracción para el modelado de interconecciones. Donde partiendo de la idea de diseño por contrato de Meyer, proponemos la noción de "**composición de contratos**" en la que se permite integración de servicios en tiempo de ejecución.

En términos generales, la **coordinación** de contratos es una conexión establecida entre objetos. A través del contratos, reglas y restricciones son superpuestas sobre el comportamiento de los objetos participantes; determinando una forma específica de interacción.

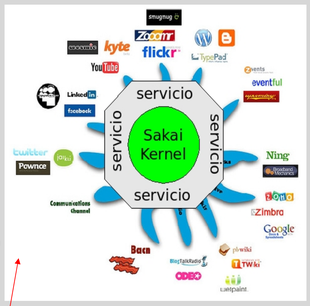




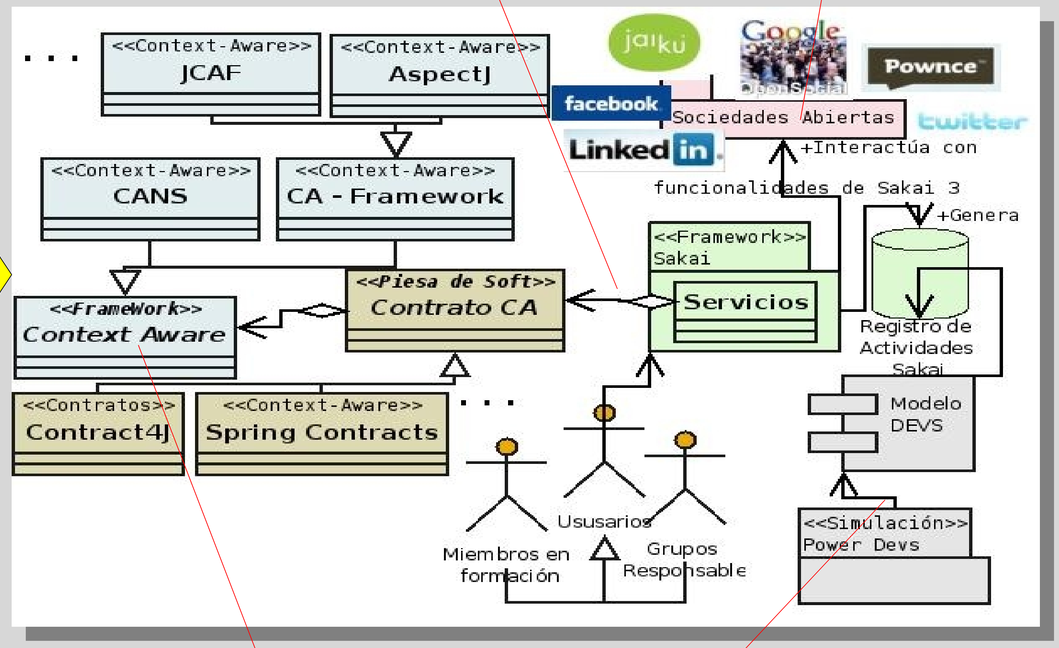
Patrón para implementar coordinación

Apache Shindig

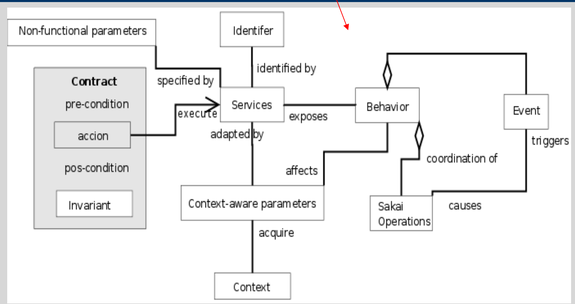
- Gadget Container JavaScript
- Gadget Server
- Open Social Container
  - Profiles, Friends, Activities, Datastore
- OpenSocial Gateway Server
  - Transferencia de Estado Representacional (REST)
  - Llamada a Procedimiento Remoto (RPC)



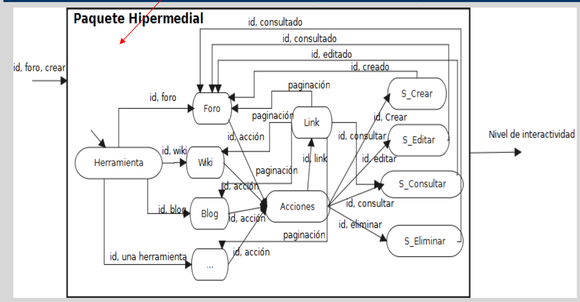
Sociedades Abiertas



Sensibilidad al contexto



Modelo DEVS del Paquete Hipermedial



Código Ejemplo

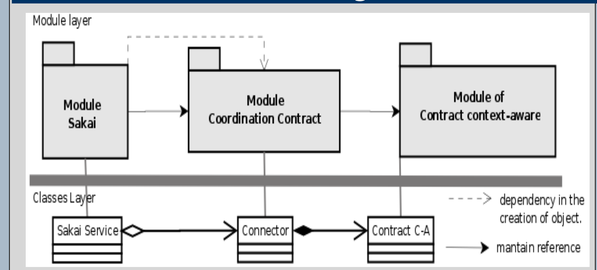
archivo2.java

```
package org.sakaiproject.discussion.impl.import java.util.* import cde.runtime.* import obab.ca.*
public abstract class BaseDiscussionService extends BaseMessageService implements DiscussionService,ContextObserver,
 EntityTransferer,ForoInterface
{
..
public messageEdit editMessage(MessageChannel channel,String id)
 {new ComponentOperationEvent(this , "Edit").fireEvent()
 return (MessageEdit )super.editResource(channel,id)
 }
//Code added by the contract generator
protected CrdProxy _proxy
//Class Ids definition
private static Class _classId-Sakai.Discussion.class
public static Class GetClassId() {return _classId;}

public CrdProxy GetProxy() {return _proxy ;}
public void SetProxy(Object p ) {
 if ( p instanceof CrdProxy && p instanceof DiscussionInterface) _proxy =(CrdProxy)p ; //else exception!
public void SetProxy(CrdProxy p) { _proxy =p ;}

AccountInterface GetProxy_Account() {
if ( _proxy == null ) return null; return ((DiscussionInterface) _proxy.GetProxy().__classId());

//These methods implement the Proxy Client part of the Subject

public long getMessage()
{ new ComponentRequestEvent(this ,"getMessage").fireEvent()
 AccountInterface proxy =GetProxy ,Discussion()
 if (proxy==null )proxy=this ;return ((getMessage() else (long value =proxy.getNumber();return value ;)
```

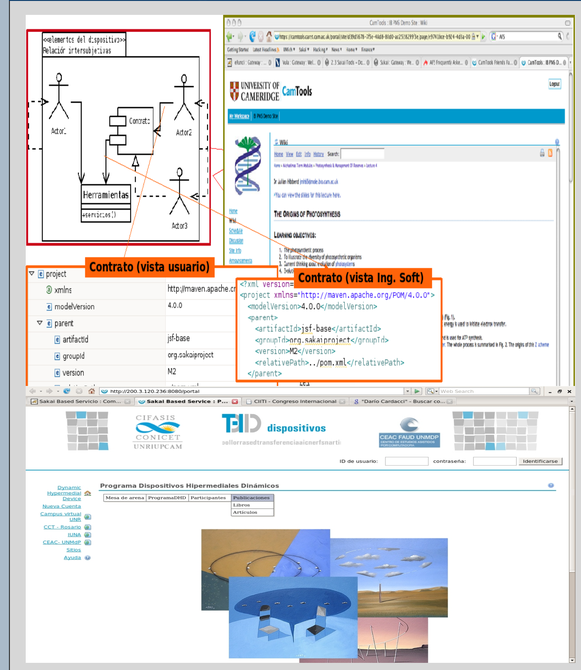
archivo3.java

```
package org.sakaiproject.import java.util.* import cde.runtime.* import obab.ca.*
public abstract class IDiscussionPartner extends CrdContractPartner implements CrdProxy,DiscussionInterface {
 ****Attributes ****;
 protected Discussion subject;
 ****Constructor ****/
 public IDiscussionPartner(CrdContract ct,Class c)
 {super(ct,c);}
 *** Definition of abstract methods in CrdContractPartner ***/
 public void SetProxy(Object p ){subject.SetProxy(p);}
 protected Object GetSubject() {return subject;}
 public void ResetProxy(){subject.SetProxy(null);}
 *** Typed Access Methods. ***/
 protected Discussion GetSubject(Discussion[]){return (Discussion)subject;}
 protected IDiscussionPartner GetNextPartner(Discussion[]){return
 (IDiscussionPartner )GetNextPartner(Discussion.GetClassId());}
 protected IDiscussionPartner GetNextPartner(Discussion[],CrdPartnerRulesListIterator _r){return
  _r.hasNext();r (IDiscussionPartner )((CrdPartnerRules) _r.next());partner null;}
 **Default Implementation of Discussion Interface **/
 public void messageEdit (double amount,Customer c)throws DiscussionException
 {
 IDiscussionPartner next =GetNextPartner(Discussion());
  if (next != null )next.editMessage(amount,c);else GetSubject(Discussion()).editMessage(amount,c);
 }
 **Default Implementation for condition checkings **/

 public CrdPartnerRules messageEdit(rules,String texto,Student c)throws DiscussionException,CrdExFailure
 {return new CrdPartnerRules (this);
```

Modelo de Integración



Configuración del contrato



POSTER TEMPLATE BY:
www.PosterPresentations.com