Semantically augmented web services brokering mechanism based on hierarchical component system

Gábor Paller, Nokia Siemens Networks
Stian Alapnes, Telenor R&I
Anthony Tarlano, DoCoMo Euro-Labs
Jussi Riihelä, Nokia Siemens Networks
2007 June 1

Presentation overview

- We present currently ongoing research in the SPICE project (IST Contract No. 027617).
- The aim of the research activity is to define a flexible brokering mechanism
  - uses semantically enhanced discovery to locate services.
  - composes service networks with high autonomy.
- hierarchical component model was developed and applied to the web services domain.
- The broker supports pluggable composition mechanism interface that provides support for unlimited number of composition methods.
- Semantic matching is supported by the Discovery Facility

wireless world initiative

SPICE platform

Service adaptation in SPICE platform

- intelligent adaptation to the
  - user’s needs
  - environment
  - business contracts
- that the best end user experience can be provided in every situation.

SPICE support of adaptive services

- Front end: Knowledge Management Framework (KMF); extensive framework for obtaining, processing, forecasting, etc. context information
- Registry: looking up services according to semantic queries: Discovery Facility
- Back end: adapting services according to KMF information: the *broker*

Requirements against the broker

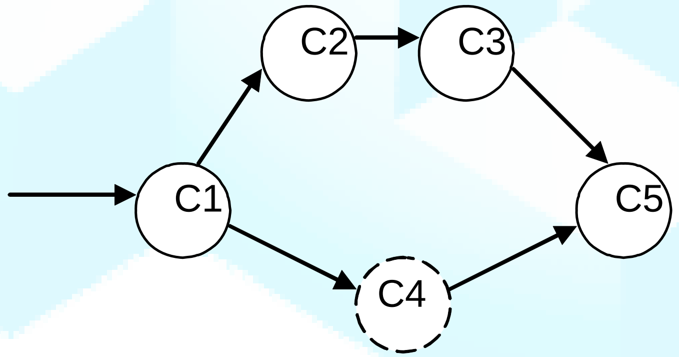
- **The broker must consider**
  - Functional and semantic annotations exposed by components,
  - Security requirements, particularly those that describe, how end user services are exposed toward end users,
  - Context information obtained from the Knowledge Management Framework (KMF).
  - Contract information exposed by components that control which components can be connected.
  - Service Level Agreements (SLA)
- **The broker must incorporate multiple composition methods**
  - the composition method field is still quite diverse
  - comes with a number of trade-offs with regards to
    - consistency of the composed service network
    - the speed of the service network calculation
    - the flexibility or “intelligence” of the service network calculation

Approach

- A hierarchical, abstract component model was designed with the explicit goal of supporting automatic composition.
- This abstract model was mapped to the web service model the broker is expected to operate in.
- The broker's architecture and algorithms are then designed that operate on this model mapped to web services.
  - Each composition mechanism supported by the broker is expected to use the same component model.
  - The usage of the elements of the component model and the way of describing the composition, however, may be very different depending on the composition mechanism.

Component model

- **Largely influenced by Fractal**
  - Basic components: cannot be decomposed further into other components
  - Composite components: may be decomposed into other components
- **A component network is said to be abstract if its components or component connections are not fully known at design time**



Adaptation rule: select C4 from components implementing mapInterfaceInput and mapInterfaceOutput that fit to the current location context.

Component interface annotations

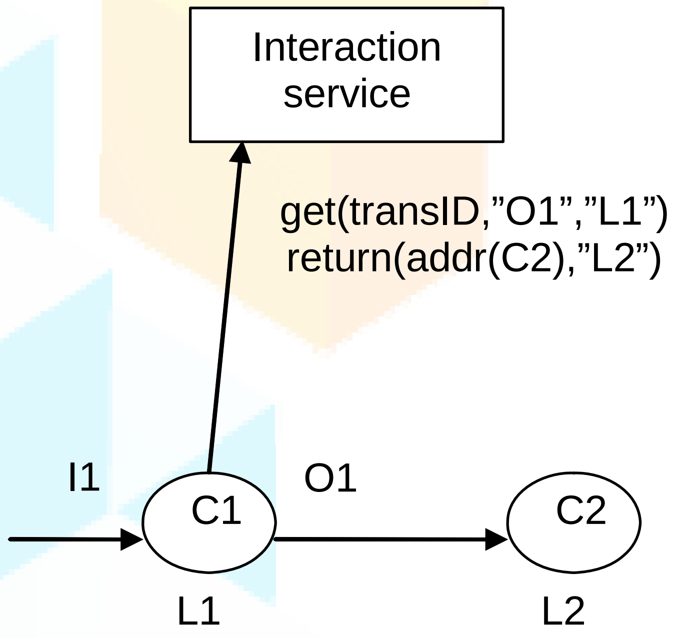
- **Component interfaces are annotated with meta-info to facilitate automatic composition**
  - Interface types: what interfaces can be connected
  - Message exchange patterns: request-response and unidirectional
  - Multiplicity: how many input interfaces can be connected to one output interface
  - Constraints: ontology-based constraint language in order to refine type information (e.g. the output is temperature but is given in Celsius)
  - Message Container Type: transport for the messages (e.g. CORBA)
  - Service Level Agreement advertisements

wireless world initiative

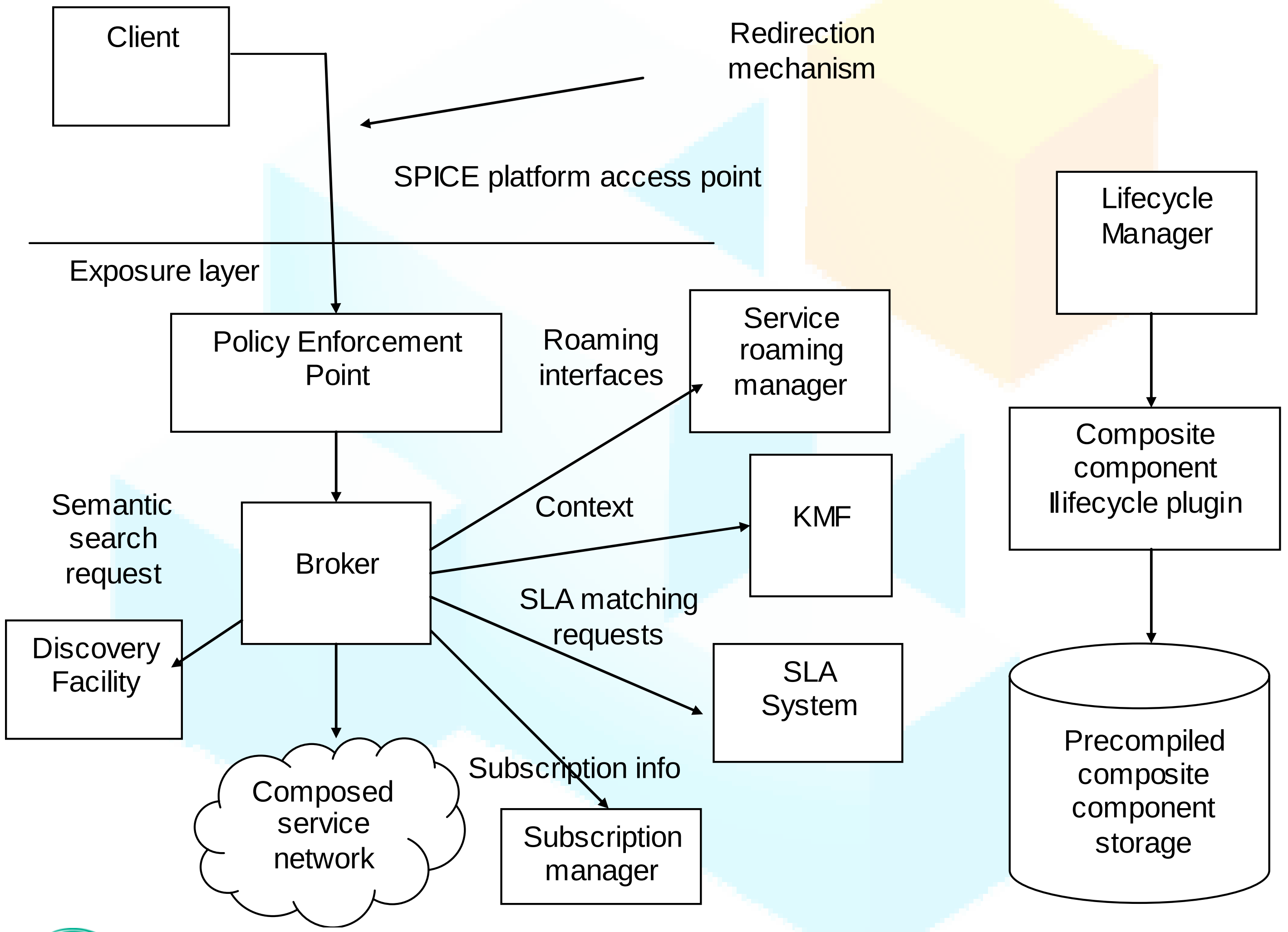
Mapping the component model to SOA

Mapping bindings

- In SOA, we don't "instantiate" components. Instead, component bindings are associated with transactions.
- Two solutions
  - the component network is propagated with the requests
  - interaction service stores the component network and each component retrieves the binding when it calls the next component in the network.



Broker from helicopter view



Discovery Facility

- **The Discovery Facility (DF) is the functional unit in the SPICE system providing a registry service for semantic acquisition of SPICE components.**
- **Using a registry model the Discovery Facility allows a service component to publish semantic and non-semantic metadata that describes a component's service capabilities**
- **allowing those capabilities to be discoverable by architectural entities within the SPICE platform.**
- **By using the Interceptor Design Pattern the Discovery Facility can be extended transparently**
  - in order to achieve a "Separation of Concerns"
  - decouples the internal operation of the Discovery Facility from the query evaluation, i.e. matchmaking
  - allowing a wide range of metadata specification technologies to be supported within the registry.

wireless world initiative

Broker/DF interaction

1. Reception of a request message containing one or more semantic queries to be evaluated over the current set of published component artifacts.
2. Parsing of the message to identify individual query expressions.
3. Classification of each query expression language (Explicit or Implicit).
4. Plug-in based matchmaker evaluation of individual query expressions over the functional and semantic annotations exposed by components.
5. Return the service component set that match the request of the broker in order for the broker to identify best match for binding in the service component network.

Brokering composite components

- **Every service request served by the service platform is handled by a composite component.**
- **Every composite component has a composition method associated to it.**
- **Composition method may be e.g. semantically enhanced BPEL.**
- **Two stages of the broker**
  - Composition method resolves the abstract service component network to a concrete service component network (all nodes and bindings are resolved)
  - Concrete service component network is then executed.

Implementation alternatives

- **Full vs. partial binding**
  - Full binding of the service component network resolves the entire component network before it is executed.
  - Partial binding delegates the resolution of dynamically selected components to service network execution time. Each dynamically selected component is replaced by a proxy component that accesses the Discovery Facility when executed.
- **Direct component-component communication vs. dedicated execution engine**
  - Execution by a dedicated execution engine. In this case the execution engine receives the service component network and acts as a central distribution point that sends requests to components and receives responses.
  - Direct component-component communication without execution engine In this case the components are connected to each other and there is no router among them.

Composition sessions

- **Service component network types**
  - Statically composed service component networks that remain the same independently of conditions in the environment.
  - Dynamically composed service component networks are composed based on environmental conditions
    - service component networks belong to users and a separate version of the service network is calculated and temporarily cached for each user.
- **The lifetime of the service component network may be one of the following.**
  - Per-request: the service component network is composed for just the duration of serving the actual request.
  - Per-session: the service component network is composed for the duration of multiple requests. This duration is called composition session to distinguish it from other sessions used in the SPICE platform.
  - Per-application: once the composite component is composed, the service component network is persistently saved and is not recalculated.
- **Events for per-session service component network invalidation:**
  - logout event, timeout event, KMF event and explicit application request.
  - KMF event as invalidation event: attentive services

wireless world initiative

Composition methods: semantic BPEL

- Uses standard extension mechanism of BPEL
- Obtains environment information from KMF and adapts DF queries accordingly

-
  ```
  <sref:service-ref><addr:DynamicEndpoint>
  <addr:KnowledgeSource ksref="loc">
  <addr:KnowledgeClassURI>http://amigo.gforge.inria.fr/owl/AmigoICCS.ow
  l#City</addr:KnowledgeClassURI>
  <addr:KnowledgeFilter>locfilter.xsl</addr:KnowledgeFilter>
  </addr:KnowledgeSource>
  <addr:EndpointFilter>
  <![CDATA[
     PREFIX spice: http://spice.example.org/services/0.1/
     PREFIX loc: http://amigo.gforge.inria.fr/owl/AmigoICCS.owl
     SELECT ?service
         WHERE
         {
       ?service spice:implements spice:mapInfo .
       ?service spice:location loc:$(loc)
     } ]]>
  </addr:EndpointFilter>
  </addr:DynamicEndpoint></sref:service-ref>
  ```

OWL-S composition

- **One of the motivating tasks of OWL-S is to provide automatic service composition and interoperation. To facilitate this, services are considered to be processes.**
- **Information handled by the ServiceModel includes IOPR (Input/Output/Precondition/ Result), which play an important role in OWL-S based composition.**
- **The OWL-S ontology defines two main types of processes:**
  - Atomic processes always have two participants, TheClient and TheServer, they take one message as input (from TheClient) and produces one output message (from TheServer).
  - A composite process maintains a state, and can handle several input messages, each input (possibly) changing the state of the process. Composite processes can be decomposed into other processes and the decomposition is specified by control constructs.
- **The ControlConstruct class has several sub classes, each specifying a construction mode: Sequence, Split, Split + Join, Choice, Any-Order, Condition, If-Then-Else, Iterate, Repeat-While, Repeat-Until, and AsProcess.**
  - Using the control constructs one can combine processes (both atomic and composite) from several sources to compose new processes and eventually services.
  - This is where the IOPR of the processes is used: When building a call graph, the IOPR of one process must match the IOPR of the next process.

wireless world initiative

Conclusion

- **Broker is a central entity of the SPICE and its features largely determine the “intelligent” features of the platform.**
- **The SPICE broker is a sensitive trade-off between advanced features, high performance and security.**
  - supports request-time dynamic composition
  - restricts request-time composition to relatively simple operations.
- **As the broker is based on a plug-in architecture, it may be used as a test bench for different composition methods while keeping the component model, hence the application code unchanged.**