(Draft) Proceedings of EUROCAST 2003: Cast and Tools for Complexity in Biological, Physical and Engineering Systems: 9th International Workshop on Computer Aided Systems Theory. 2003, p. 237-239.

# Generative Communication with Semantic Matching in Distributed Heterogeneous Environments*

Pedro Álvarez[1], José A. Bañares[1], Eloy J. Mata[2], Pedro R. Muro-Medrano[1], and Julio Rubio[2]

[1] Department of Computer Science and Systems Engineering, University of Zaragoza, María de Luna 3, E-50015 Zaragoza (Spain).
{alvaper,banares,prmuro}@posta.unizar.es

[2] Department of Mathematics and Computer Science, University of La Rioja, Edificio Vives, Luis de Ulloa s/n, E-26004 Logroño (La Rioja, Spain).
{eloy.mata,julio.rubio}@dmc.unirioja.es

**Abstract.** Different standard middleware proposals have emerged to provide computing models and communication among components in open distributed systems. Nowadays, Internet is becoming an increasingly relevant alternative to middleware platforms, due to the success of Web services in solving problems of application-to-application integration in distributed and highly heterogeneous environments. However, a coordination model is necessary to build open and flexible systems from active and independent distributed components. In this paper, we present a Web-enabled Coordination Service to orchestrate heterogeneous applications based on the Generative Communication model with semantic matching. Our aim is to use Internet as a real distributed computing platform, considering heterogeneous semantic interoperability.

## 1 Introduction

Traditional middleware platforms (such as CORBA, COM or EJB) are sometimes presented as a general solution for distributed computing in heterogeneous contexts. Nevertheless, this is not completely true in practice. On one hand, they are based on object-oriented constructs and then some degree of homogeneity is required, at least from a programming-paradigm point of view. On the other hand, the physical substrate, on which communications are established, is abstracted. (Note that these two features of middleware are not considered negative for us; they simply imply certain consequences that are not always explicitly stated when these general platforms are introduced.) Internet is becoming an increasingly relevant alternative to standard middleware platforms, due to the success of Web-services in solving problems of application-to-application integration in distributed and highly heterogeneous environments. These Web-services

* Partially supported by the Spanish Ministry of Science and Technology through projects TIC2000-1568-C03-01, TIC2000-0048-P4-02, TIC2002-01626.

may be implemented on different computational models, and communicate and interchange data among them using standard Internet protocols and data formats, such as HTTP and XML, respectively. However, there is no standard support enabling these distributed services to work together harmoniously and in a *coordinated* way (this is one of the main difficulties inherited from traditional approaches). To enable this cooperation among distributed services it is necessary to take into account two essential aspects of Internet. First, from a technical point of view, communication by means of HTTP is always synchronous (via sockets, for instance), but in distributed and concurrent computing asynchronous communication is also mandatory. The second aspect has to do with the *real* behaviour of Internet: it is a hostile medium where the reliability of the communications is poor and unsafe.

In this paper, a proposal to overcome some of these difficulties, by means of a Web-services based approach, is presented. Our aim is to use Internet as a real distributed computing platform, considering semantic interoperability among systems developed in very different programming languages, different even from a programming paradigm point of view. (It is worth noting that we do not consider our proposal as an alternative to standard middleware; it is rather an experience linked to a set of ideas largely orthogonal to any particular implementation tool.)

The basic idea is to use a *coordination model* that acts as a kind of "glue" gathering separated activities into a single computing device. Instead of starting from scratch, the model known as Generative Communication has been chosen, and more precisely the so-called *Linda* model [4]. Linda is a very abstract artifact based on two notions: tuple and tuple space. The tuples are extracted from the tuple space by means of a pattern matching process. Our proposal is based on this obvious observation: if the simple matching strategy of Linda is replaced with a *complex matching*, then very general kinds of interoperability can be achieved. To this aim, we work with a version of Linda where the tuples admit a description by means of attribute/value pairs, which is similar to a XML-data definition, extracted from [5]. Now, Linda can be used in this XML context, providing a promising way to coordinate, communicate and collaborate on the net.

These ideas, which can be summarized as "Linda with semantic and structured matching, using XML", have been put into practice to deal with Web-services, but trying to avoid the two difficulties previously mentioned: the synchronous and hostile nature of Internet. Our implementation uses JavaSpaces as core. (JavaSpaces is a Linda realization based on the Jini technology.) This saves us the rewriting of routine code to manage tuples and tuple spaces. We have enriched JavaSpaces to work with XML-entries. Nevertheless, note that JavaSpaces only provides the "top level", non-structured (and, of course, non-semantic) matching. From this extension of JavaSpaces, a Web Coordination Service has been developed, achieving the following objectives:

1. Coordination operations are accessible through the HTTP service interface, using a XML data format to specify the exchanged messages.

2. We have obtained a uniform way to deal with complex matching (both semantic and structured).
3. We have included an event-based asynchronous communication over HTTP, generated through PushLet technology. This communication style has allowed us to incorporate reactive coordination aspects to the Generative Communication model (by means of a system of subscriptions).
4. We have created internal agents/proxies to represent the external Web-services and cooperate on behalf of them. This reduces the number of Internet connections and hence minimizes the problems of reliability.

The paper is structured as follow. Section 2 shows a model for coordinating open Web services. Section 3 presents a description of the Web-enabled Coordination Service (WCS). The WCS consists of three software components: XML-based Space, Java Coordination Component and HTTP Coordination Component. In Section 4, the semantic pattern matching is described. The matching is made in two steps and some semantic resources, such as thesauri and ontologies, are used. Section 5 tries to formalize the operations of the coordination model and the complex matching process of our approach. In Section 6 we show two application examples: the Dinning Philosophers problem, as an academic example; and a real project in the context of location-based services and automatic vehicle-monitoring (to orchestrate different OpenGIS and LIF Web services). Finally, conclusions and future work are presented.

## 2 A model for coordinating open Web services

Web Services are self-contained, modular applications that can be described, published, discovered and invoked over a network. Then, the Web service based approach is an application integration concept, and some essential elements should exist to support it:

– Component interfaces must be *Web-enabled*, providing a collection of functions that are packaged as a single entity and published to the network to be used by other programs. These functions must be specified according to wide-accepted standards that ensure interoperability, ease of use, and loose coupling of Web services.
– It is necessary a universal *middleware for interchanging data* that ensure interoperability beyond specific distributed computing platforms (e.g., COM, CORBA, EJB) and/or different programming languages. Web services communicate using HTTP and XML. Therefore any device, which supports these technologies, can both provide and access Web services.
– A *coordination model* that acts as the glue that binds separate activities into an ensemble. Distributed applications and Web services are a natural breeding ground for heterogeneity. A service that needs contributions from different machines, providers and computing models requires a natural means for multi-language. Therefore, it is necessary to provide a coordination model to represent these interactions and the space where they happen.

In this context, each Web service can be seen as a type of individual entity, which needs to communicate and synchronize with another Web services, or entities distributed over Internet to get together a response for the user's request. It is necessary to emphasize that Internet is a highly heterogeneous cooperation environment. Many Web services are developed in different programming languages, are running over different executing environments or/and are a part of a framework designed under the assumption that it is fully in control of the execution loop and developed with different architectural styles to ensure their usefulness in specific contexts [9].

Therefore, the coordination model to represent interactions among Web services must be orthogonal regarding this heterogeneity. The *Generative Communication model*, alternatively Tuple Space Communication model, provides the illusion of a shared memory, called *Tuple Space*, to allow that the inter-process communication was uncoupled logically, temporarily, and spatially by means of *tuples*. A *tuple* is something like `["Gelernter",1989]`, where the components are supposed to be untyped, atomic values. The best-known example of coordination language based on Generative Communication is *Linda* [7]. Linda provides four basic operations: `eval` and `out` to create and insert new tuples into the tuple space; `in` to read and remove at the same time a tuple from the tuple space; and `rd` to read a tuple without removing it. Using these operators, sender and receiver processes cooperate among them in an uncoupled way. Linda is a model of process creation and coordination that is orthogonal to the base computation language in which it is embedded. It does not care how the multiple execution threads in a Linda program compute information; it deals only with how these threads have been created, and how they can be organized into a coherent program.

## 3 Description of the Web-enabled Coordination Service

We have implemented a *Web-enabled Coordination Service* (WCS) to support the coordination among Web services. The coordination language Linda has been chosen to model the coordination functionality. The WCS is a network-accessible software that provides a set of services to communicate and to synchronize heterogeneous applications distributed over Internet. It offers highly uncoupled single or group communication services. These are data storage services that can hold them beyond the life of the generating distributed-applications and event notification services. Every distributed application, regardless of the hardware and operating system platforms where they are running, the programming language used to encode them and the middleware itself, must be able to gain access to these service to cooperate among them. The selected approach is through open Internet protocols, such as HTTP, and using standard-data format to encode exchanging data, such as XML.

As it is shown in the Fig. 1, WCS is composed by three software components that provide the previously presented services. Their roles and responsibilities into the coordination service are presented:

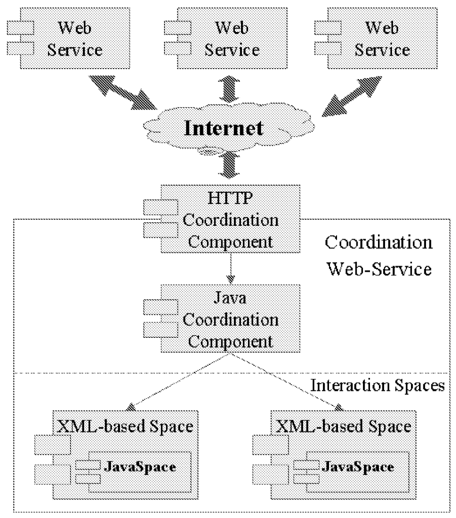


**Fig. 1.** Web-enabled Coordination Service.

**XML-based Space.** This component has been developed based on JavaSpaces technology, and encapsulates inside an interaction space where a collection of processes through the interface of this component can cooperate among them exchanging XML documents. Besides, these XML documents can be defined in execution time. We have extended Linda with the notion of structured tuples which allows us representing XML documents as lists of attribute/value pairs. These XML documents are stored into the interaction space as Java objects.

The XML-based Spaces component tries to provide a more flexible way of working than JavaSpaces. It is because XML can be used to describe everything in a simple, powerful, and easy to understand way.

**Java Coordination Component.** This Java component is the core of the coordination service. It provides a collection of coordination operations, defined from the Generative Communication model previously presented, that are divided into two different interfaces called the *Basic Coordination* (BCI) and the *Reactive Coordination* (RCI) *interfaces*. The BCI offers a set of simple communication and synchronization operations among processes. These basic operations encourage a programming style where processes that invoke an operation can block until a communication is completed or a synchronization condition is achieved. This style could not be the most adequate to coordinate distributed processes. A reactive programming style based on distributed events can be used to design and build the coordination among distributed processes. To support this reactive style, the RCI provides operations so that a process can advertise its interest to generate a specific type of events, publish the advertised events and subscribe its interest to receive events of a specific type.

**HTTP Coordination Component.** This component plays a role as a Web accessible interface of the coordination component previously presented. It provides, through its two interfaces, called *HTTP Basic Coordination Interface* (HBCI) and *HTTP Reactive Coordination Interface* (HRCI), the same collection of operations than the Java Coordination Component. These interfaces describe coordination operations that are Web-accessible through Web protocols and data formats, such as HTTP and XML.

The nature of these interfaces hides the implementation details of the service so that it can be used for another Web-applications independently of the hardware and software platform where they are running and independently of the programming language in which they are written. This approach combines the best aspects of component-based development and the Web, and it is the cornerstone of the Web-Service-based model [6]. This model allows and encourages Web-Service-based applications to be loosely coupled, component-oriented and with cross-technology implementations.

## 4 Pattern matching and semantic disambiguation

According to the LINDA model, the operation `in(x?)` tries to match the tuple `x?` with a tuple in the shared space. If there is a match, the tuple is extracted from the tuple space; otherwise, it blocks until a convenient tuple appears. The parameter for `in()` can be a *query tuple* with a wildcard, like in `["Gelernter",???]`. The match is then *free* for the wildcard and *literal* for the constant values. Our proposal is based on this obvious observation: if this simple matching strategy is replaced with a *complex matching*, then very general kinds of interoperability can be achieved.

Considering this basic idea let us particularize the concept of *complex matching*. To this aim, we work with a version of Linda where the tuples admit a description by means of attribute/value pairs, like:

`[(author,"Gelernter"),(year,1989)].`

Although this is still an untyped setting, this *bit* of structure allows us recovering information from a distributed context. Thus, if an operation `in()` is invoked in a different institution, where the term "author" is not used, but "creator" is used in its place, and if there is a convenient mapping between ontologies, then the request `[(creator,???),(year,1989)]` can be successfully satisfied. This type of *semantic matching* has already been exploited by our team in the context of GIS interoperability [8], and can be used to support multilingual interoperability. Interestingly enough, this semantic mechanism is implemented through Internet, using XML as transfer format.

Let us consider the following XML-data definition, extracted from [5].

```
Xml = (union String (cons Symbol (cons Att LXml)))
LXml = (listof Xml)
Att = ...
```

In that paper, this definition was implemented in Scheme, but it is clear how to directly translate it to languages as ML or Haskell, or, with a little more effort, to any other programming language. Anyway, the important remark for our current presentation is that the "top level" structure of any XML document admits the expression

```
[(att1,<val 1>),...,(attN,<val N>)],
```

but where each `<val i>` is structured (in particular, it can be XML-based). Now, Linda can be used in this XML context, providing a promising way to coordinate, communicate and collaborate on the net.

To achieve the semantic pattern matching we must face previously with the problem of word sense disambiguation. This problem is perhaps the greatest existing problem at the lexical level in natural language processing [12], and this skill is applicable to tasks such as information retrieval, machine translation, speech synthesis and pattern matching. The problem of disambiguation consists in determining which one of the senses of an ambiguous word is invoked in a particular context composed of a set of words related to the ambiguous word. A word is ambiguous (or polysemic) if its sense changes depending on the context.

There are different statistical approaches to solve this problem depending on the training material available. Supervised disambiguation methods are based on a previously disambiguated training set. Each occurrence of the ambiguous word is annotated with its contextually appropriated sense. Based on this information, the aim of these methods is to build a classifier, which correctly classifies new cases. On the other hand, unsupervised disambiguation methods try to distinguish among the senses of a polysemic word without help of disambiguated examples. Unsupervised methods are based only on the features that can be automatically extracted from unlabeled texts. There are also methods that use lexical resources such as machine-readable dictionaries, lexical knowledge bases or thesauri. These methods rely on the definition of senses in dictionaries and other lexical resources. Sometimes they are merged with training supervised or unsupervised methods.

The disambiguation method that we developed [8] can be considered as an unsupervised disambiguation method based on the hierarchical structure of WordNet and the notion of conceptual distance among concepts. The WordNet ontology is organized around the notion of *synset*, that is, set of synonyms that expresses a concept. By means of a voting system, our method takes a term from a thesaurus, where the terms are organized in hierarchical structures similar to trees whose nodes are the terms which maintain associations with their broader (ascendants) or narrower terms (descendants), and it tries to determine the "closest" sense (WordNet synset) to the senses of the other words in the whole branch that constitute its context.

As it has been previously mentioned, the XML-based Space component is based on JavaSpace Technology. It provides an object space for storing these XML-tuples, avoiding the development of a new XML-tuples space. However, the matching rules of JavaSpaces are not adequate for working over objects with structured fields. Objects are matched by complete fields, not within the contents of a field. This is owing to objects are serialized for storing them into the space, and the matching between objects is made applying an equality operator on the corresponding field value. In the case of a structured field, its value is the serialization of each component. The generic Java object that encodes any XML-tuple has structured fields to store the nodes of the XML tuple. This generic object has two structured fields to store the tag names and values of the XML-tuple. The tag name of the first node of the XML-tuple is stored in the first component of the tag-name field and its value in the first component of the tag-value field, and so on.

Therefore, it is necessary to increase the rules of JavaSpaces to allow the matching by the contents of a structured field. To resolve the rule restrictions, the matching is made in two steps. In the first step, the matching rules of JavaSpaces are used. The original template is saved for the second step and a copy of it is created for being used in the first step. The tag-value field of this copy is set to the null value (see our tentative of formalization in the next section). When it executes a read operation (provided by the JavaSpace interface) using this template object, a returned object represents an XML-tuple with the same XML-Schema as the template because the tag-name field is only considered for the matching. In a second step, it is invoked a particular matching method of the retrieved object, using the original template as a real parameter. The method checks that each not-null component of the template's value-field have the same value in the corresponding component of the retrieved object's value-field. If it returns a true value, the retrieved object matches the template according the XML-tuple matching rules. Otherwise, the retrieved object has the same XML-Schema as the template but it does not match the template according to the second matching rule for XML-tuples. Then, the first step is made again until an object matches according the XML-tuple matching rules.

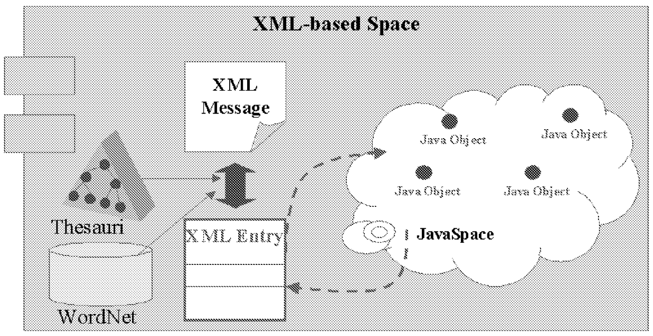


**Fig. 2.** Semantic resources.

If the words of the tag-name fields or of the tag-values fields have been extracted from a thesaurus, then it is possible, in the second step of the matching process (see Fig. 2), to make use of lexical and semantic resources, such as

(Draft) Proceedings of EUROCAST 2003: Cast and Tools for Complexity in Biological, Physical and Engineering Systems: 9th International Workshop on Computer Aided Systems Theory. 2003, p. 237-239.

WordNet ontology and, by means of our disambiguating method, we can enrich the XML-based Space component of WCS with a semantic pattern matching where the retrieval of a XML document is based on the structural and semantic similarity of a document with a given template.

## 5 Towards a formalization of our approach

When applying formal methods to model real-life systems, several degrees of abstraction can be used, depending on the objectives the analyst is looking for. In the *coordination* area, there are extremely abstract approaches as those of [2] or [3], where the formalisms for tuple-based coordination are based on process algebras. Taking into account that one of the main features of this Linda-like coordination is the associative access to a shared memory, the algebraic views in [2] or [3], where in particular any tuple space consideration is abstracted, could be considered too unrealistic. However, this has not to be observed as an inadequacy, but rather as a precise choice in order to analyze, in a way as simple as possible, some theoretical characteristics of the models (such as its expressiveness, for instance)

Other authors, as [10] or [11], having in mind different goals (as the analysis of event-based coordination), have presented formalisms in which tuple spaces are explicitly represented, and coordination gives rise to a lower level model (with respect to [2], [3]) by means of transition systems. However, in these approaches the matching process is completely abstracted through a *matching predicate*, considered as predetermined. It is quite clear that this approach is also too abstract to model our proposal.

In order to find the right *abstraction grain* for our task, three components appear as clearly distinguished. The first one is that our coordination model is constructed on top of another, more basic and predetermined, coordination artifact as JavaSpaces. The second one is that templates are structured and they must be handle in order to enable semantic XML-based interoperability. The third one is that the final coordination-service behaviour should be expressed in terms of the two previous points, and not in terms of some ad-hoc implementation details. Let us comment briefly each one of these three points.

In a first approach, we could replace the complex features of JavaSpaces by a formal model for Linda, for instance that of [11]. The main ingredients of this model (see [11] for details) are a set of tuples $T$; a set of templates $Templ$; a matching predicate $mtc(templ, t)$, where $templ$ ranges over $Templ$ and $t$ over $T$; a choice operator $\mu(templ, \tau)$ extracting a tuple $t \in \tau$ which matches a (multi)set of tuples $\tau$ or returning an error element $\perp$ if no matching is available; and the standard Linda operations as `in`, `rd`, `out` and so on. Since this will be integrated as a *core* coordination model, auxiliary to define another one, let us denote each ingredient with the subindex *core*: $mtc_{core}$, $\mu_{core}$, $in_{core}$, etc.

With respect to the complex matching process, it is necessary to give more (mathematical) structure to the templates set $Templ$. It is clear that templates are sorted with respect to generality; for instance, `["Gelernter",???, ???]`

is more general than `["Gelernter",???, 1988]`. This relation can be axiomatized: $templ_1 < templ_2$ if $\forall t \in T_{core}$, $mtc_{core}(templ_2, t) \Rightarrow mtc_{core}(temp_1, t)$. Thus, $Templ$ is endowed with a partial order, with one minimal element for each arity of tuples; explicitly, the minimal elements are `[????]`, `[???,???]`, ...

This scenario can be generalized if we assume that templates and tuples occur as sequences of pairs `(attribute,value)`. Then, if wildcards are only allowed as values, other minimal templates appear:

`[(author,???)],(title,???),(year,???)]`

Let us introduce an operation[3] $min : Templ \rightarrow Templ$ associating to each template its minimal associated template.

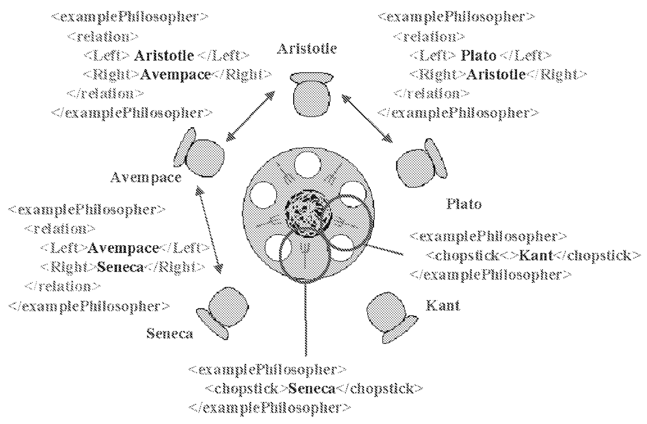
In order to define the final coordination service in top of the two previous elements, at least two approaches are possible. In the first one, we considered not only the tuples in $T_{core}$, but also we consider other *virtual* tuples $T_{virt}$ that can be constructed canonically from $T_{core}$ by means of some disambiguation process (a typical example would be the case of a multilingual coordination service where, with the help of *dictionary mappings*, "virtual" tuples in other languages are considered). In this case, the new tuple space to be defined would be: $T_{def} := T_{core} \cup T_{virt}$. Then, the matching predicate $mtc_{def}$ could be simply defined from $mtc_{core}$, but the choice operator, $\mu_{def}$, would be in charge of constructing the possibly virtual counterpart of the tuple recovered by $\mu_{core}$. The alternative is to define simply $T_{def} := T_{core}$, and then the burden of working with the semantic aspects is for both $mtc_{def}$ and $\mu_{def}$. In any case, a possible definition of the operation $in_{def}$ could be sketched as follows. Giving a query $in_{def}(temp)$, the minimal template $min(temp)$ is used to query the underlying Linda Model as $rd_{core}(min(temp))$, in order to do no interference with other open processes. If $\perp$ is obtained (in other words, if the internal query blocks), this is the same for the original query (by definition of $min()$). Otherwise, a tuple is obtained and now can be processed in order to know (by means of $mtc_{def}$) whether it satisfies the original query; if it succeeds the tuple recovered by $rd_{core}(min(temp))$ is then removed.

Obviously, this very brief presentation is not only incomplete, but also let many points to be studied, in particular related to the soundness and fairness of the new defined coordination service. Nevertheless, we think that the main pieces are here established in order to analyze formally the theoretical properties of the Web coordination service proposed and, in particular, to prove if it accomplishes the very definition of some Linda-like coordination model.

## 6 Application examples

We have showed the applicability of the WCS with a classical problem. We have implemented a variant of the classical concurrent programming problem of the Dinning Philosophers (see Fig. 3). In our approach, several distributed philosophers, implemented in different programming languages (Lisp, Java, or HTML with JavaScript), cooperate on Internet (via HTTP and XML).

[3] This is an convolutive operator: $min(min(templ)) = min(templ)$.



```
<examplePhilosopher>
  <relation>
    <Left> Aristotle </Left>
    <Right>Avempace</Right>
  </relation>
</examplePhilosopher>
```

```
<examplePhilosopher>
  <relation>
    <Left> Plato </Left>
    <Right>Aristotle</Right>
  </relation>
</examplePhilosopher>
```

```
<examplePhilosopher>
  <relation>
    <Left>Avempace</Left>
    <Right>Seneca</Right>
  </relation>
</examplePhilosopher>
```

```
<examplePhilosopher>
  <chopstick<>Kant</chopstick>
</examplePhilosopher>
```

```
<examplePhilosopher>
  <chopstick>Seneca</chopstick>
</examplePhilosopher>
```

**Fig. 3.** The Dinning Philosophers problem.

When a philosopher is inserted, Aristotle for example, his chopstick is introduced by the `out` operator:

```
http://bubu.cps.unizar.es/CoordinationServlet?
REQUEST=<?xmlversion="1.0"?> <CoordinationService>
    <function>out</function>
    <tuple>
        <examplePhilosopher>
           <chopstick>Aristotle</chopstick>
        </examplePhilosopher>
    </tuple>
</CoordinationService>
```

and it is also introduced the relation that represents who are on the right and on the left. It supposes to recover any neighborhood relationship in order to replace it by new relations when the new philosopher is introduced. The `in` operation with the tuple `[(Left,???),(Right,???)]` recover, for example, the relationship `[(Left,Plato),(Right,Avempace)]`. The recovered relationship is replaced by tuples representing that Aristotle is on the left of Avempace, and on the right of Plato. With these tools it is easy to describe the complete process.

Furthermore, the WCS model has been the conceptual base for the development in a real problem: the Location-based services (LBS) frameworks whose functionality may be integrated into end-applications through Internet, such as ERP or CRM systems [1]. LBS frameworks require the integration of Geographic Information services, location services and communication services. Required services are built according to the Web-service approach: their operations are provided through a standard, published interface to ensure interoperability, and are accessible via HTTP and XML.

## 7 Conclusion and further work

In this paper we have presented a Web-enabled Coordination Service to orchestrate heterogeneous applications based on the Generative Communication model and implemented using Java and Internet technologies, such as HTTP and XML. It is an alternative to service-oriented architecture interaction model and independent of distributed object computing middleware. The coordination functionality of the service provides space and time uncoupling and represents an opportunistic strategy to use Web services. Furthermore, it integrates thesauri and ontologies to provide a semantic matching.

Open research issues are: (1) to work on dynamically discovering and intelligent chaining of services; (2) to extend the matcher to improve the semantic interoperability among Web services; (3) to incorporate daemons to Web services for supporting a reactive behaviour; and (4) to complete the formalization of our proposal.

## References

1. P. Álvarez, J.A. Bañares, P.R. Muro-Medrano, and F.J. Zarazaga, *Integration of location based services for field support in CRM systems*, GeoInformatics **5** (2002), no. July/August, 36–39.
2. N. Busi, R. Gorrieri, G. Zavattaro, *On the expressiviness of Linda coodination primitives*, Information and Computation 156(1-2) (2000) 90–121.
3. N. Busi, R. Gorrieri, G. Zavattaro, *Proccess Calculi for Coordination: Frame Linda to JavaSpaces*, Lectures Notes in Computer Science 1816 (2000) 198–212.
4. N. Carriero, D. Gelernter, *Linda in context*, Communications of the ACM 32 (1989) 444–458.
5. M. Felleisen, *Developing Interactive Web Programs*, Summer School on Advanced Functional Programming, 2002. To appear in Lecture Notes in Computer Science.
6. R. T. Fielding and R. N. Taylor, *Principled design of the modern Web architecture*, ACM Transactions on Internet Technology, **2** (2002), no. 2 115–150.
7. D. Gelernter, *Generative communication in Linda*, ACM Transactions on Programming Languages and Systems **7** (1985), no. 1, 80–112.
8. E. Mata, J.A. Bañares, J. Gutiérrez, P.R. Muro-Medrano, J. Rubio, *Semantic disambiguation of thesaurus as a mechanism to facilitate multilingual and thematic interoperability of geographical information catalogues*, in Proceedings 5th AGILE Conference on Geographic Information Science, Universitat Illes Balears (2002) 61–66.
9. M. Mattsson, J. Bosch, E. Fayad, *Framework Integration. Problems, Causes, Solutions*, Communications of the ACM 42 (1999) no. 10, 81–87.
10. A. Omici, E. Denti, *From tuple spaces to tuple centres*, Science of Computer Programming 41 (2001) 277–294.
11. M. Viroli, A. Ricci, *Tuple-Based Coordination Models in Event-Based Scenarios*, in Proceedings of the IEEE 22nd International Conference on Distributed Computing Systems (ICDCS 2002 Workshops) - DEBS'02 International Workshop on Distributed Event-Based Sytems - July 2002, Vienna, Austria, IEEE, 2002
12. P. Resnik, D. Yarowsky *A perspective on word sense disambiguation methods and their evaluation*, in ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What and How?, Washington, D.C. (1997) 79–86.