PadSpace: A Software Architecture for the Cooperation of Visual Components

D.Lkhamsuren and Y.Tanaka
Meme Media Laboratory,
Department of Information Science and Technology,
Hokkaido University, Sapporo 060-0814, Japan

1. INTRODUCTION

In this paper we will propose an extension of a typical Linda-like coordination model to provide mechanisms to cooperate composite visual components. This new Linda-like coordination model is called PadSpace. It introduces an extended matching functionality for nested tuples and richer data types for fields, including objects and XML documents. PadSpace use three new pads, i.e., ProviderPad, RequesterPad and ProxyPad. PadSpace has been developed as an extension to the IntelligentPad System. IntelligentPad represents information and processing components as 2D visual objects called pads.

In IntelligentPad, Each pad has both a standard user interface and a standard connection interface. PadSpace can provide a connection interface (logically) between two or many composite pads by using tuple matching mechanism of a Tuple Space for the matching of slots between two pads. In our approach, a tuple is a finite sequence of slots taken from a composite pad. Each slot consist a slot value, a slot type, a slot name and a slot path.

The main contribution of PadSpace is the mechanism that enables end-users to distribute/share their own local resources over the Tuple Space, and to use those shared resources in combination with their own resources through the automatic matching between function providers and function requesters.

2. THE MECHANISMS OF PADSPACE

Suppose an investor in Japan wants a way to look up the stock price of US-quoted companies, but to see them in Japanese yen rather than dollars. The investor holds own local company stock quote composite pad that was created by C3W[1] Framework and clipped from the CNN Money site (money.cnn.com/markets/). Specially, the CNN Money site offers a stock-price lookup in dollars.

On the other hand, another investor holds an own local currency conversion composite pad that can certainly deal with dollars and yen, and was also created by C3W Framework and clipped from the Yahoo! currency conversion page (finance.yahoo.com/m3?u).

Finally, the first investor wants to use second investor's currency conversion composite pad in combination with his own company stock quote composite pad. We describe how PadSpace's mechanisms allow these investors to share and to use these existing visual components for these purposes.

2.1. Linda Like Coordination Model

In Linda model, a tuple is a finite sequence of typed fields. We extend the field type, with a special field, named *Slot*. The *Slot* type is structured type that contains a slot value, a slot type, a slot name and a slot path, and denoted by *<slot path, slot name, slot type, slot value>*, also denoted in the compact form s.

We support the two kinds of matching method. *Slot- matching* performs matching value and type equivalence when field template contains *actuals* (containing a typed value). *Type-matching* performs matching only type equivalence when field template contains *formals* (containing only a type).

We now formalize this Linda-like coordination model extended with new *Slot* type.

Let *Slot*, including $s, s_1, \ldots,$ be a denumerable set of slot. Let *Tuple*, including $t, t_1, \ldots,$ be the set of finite sequences of slots taken from composite pad and denoted by $< s_1; \ldots; s_n >, n \geq 1$.

Templates may use wildcards to let some field unspecified; syntactically, a wildcard is denoted by *"type"* (which we assume to perform *Type-matching*). In the following, $st, st_1, \ldots,$ range over $Slot \cup \{type\}$. Finally, let *Template*, including $tm, tm_1, \ldots,$ be the set of finite sequences composed of data and wildcards denoted by $< st_1; \ldots; st_n >, n \geq 1$. We define the matching rule as follows.

Definition of Matching rule: Let $t =< s_1; \ldots; s_n >$ be a tuple and $tm =< st_1; \ldots; st_m >$ be a template; we say that *t matches tm* if the following conditions hold:

- $m = n$,
- $st_i = s_i \vee st_i = \{type\}, 1 \leq i \leq m.$

[1] C3W framework *(Clipping, Connecting, and Cloning on the Web)* to support the recombination of components of Web documents and accessing the desired information. It provides a mechanism to clip portions of Web documents, embed them into an existing Web document, and define data linkage among them by using spreadsheet like formulas.

We also separate (logically) the tuple spaces in which pads of agents interact. Essentially, we extend the tuple structure, with an attribute, named *partition*. Let P be a denumerable set of partition attribute values (including $p, p_1, \ldots,$). A tuple t, and a template tm with the partition attribute, are defined as follows:

$t =< s_1; \ldots; s_n >^p$ and $tm =< st_1; \ldots; st_n >^{pt}$, are also denoted in the compact form $t =< \vec{s} >^p$ and $tm =< \vec{st} >^{pt}$, respectively.

Definition of Matching rule with partition attribute:

Let $t =< \vec{s} >^p$ be a tuple, and $tm =< \vec{st} >^{pt}$ be a template; we say that t *matches*^{p} tm when the following conditions hold:

- $p = pt$,
- t *matches* tm

2.2. Distribution

Our support for distributing/sharing the existing composite pad to Tuple Space is implemented by ProviderPad, in Creator mode. The ProviderPad supports the extraction of slot list from the given composite pad, and also the users generate the tuple by selecting and renaming slots from that extracted slot list. Finally, users output that generated tuple to the "PadRepository" sub tuple space.

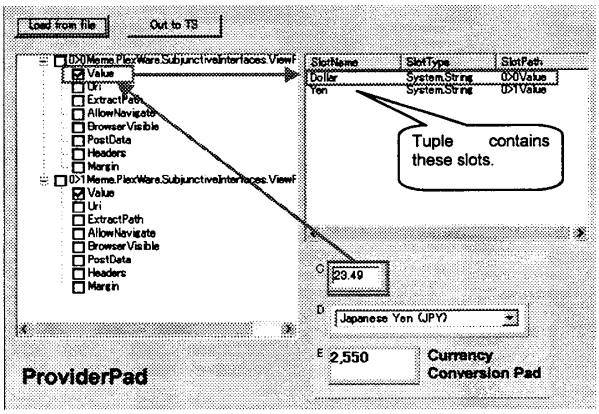

Figure 1. Tuple generation process using by the ProviderPad, in Creator mode

Figure 1 shows the user distributing/sharing the Yahoo! currency conversion composite pad.

The following is a tuple of Yahoo! currency conversion composite pad:

<Field value="23.49" type="System.String"
path="DerivationPad[0]->ViewPortPad[0]->Value">
Dollar </Field>
<Field value="2550" type="System.String"
path="DerivationPad[0]->ViewPortPad[1]->Value">**Yen**
</Field>

2.3. Consuming and Matching tuple

Our support for consuming/matching tuple from Tuple Space is implemented by RequestPad, in Matching mode. Users can build matching template using RequestPad and will be able to perform the matching with "PadRepository" sub tuples. The above sample tuple matches with the following templates:

- $rd((Dollar, Yen)^{Pad\ \mathrm{Re}\ pository})$
- $rd((Dollar, ?System.String)^{Pad\ \mathrm{Re}\ pository})$.

The RequestPad will change its working mode to Request mode and automatically load the ProxyPad as its child pad, if "t *matches*^{p} tm" is successfully completed. After the loading of ProxyPad, it adds *Dollar* and *Yen* slots to the own slot list and outputs a connection request tuple to "Connection" sub tuple space.

2.4. Communication

The ProviderPad will change its working mode to Provider mode and load the source composite pad (Yahoo! Currency conversion pad) as the own child pad, when it has received the connection request from ProxyPad. The source pad and ProxyPad make a slot connection with each other through "DataExchange" sub tuple space. The ProxyPad outputs a currency conversion request tuple to "DataExchange" sub tuple space, when it has received "set" message from its any slot. The ProviderPad receives this request tuple using "t *matches*^{p} tm" and transmits slot values to the source pad. After performing the source pad's internal process, the ProviderPad outputs the currency conversion result tuple to "DataExchange" sub tuple space.

2.5. Multiple Matching

We also support Multiple Matching functionality. For example first investor wants to compare two conversion results www.gocurrency.com and finance.yahoo.com/m3?u. In Multiple Matching, the ProxyPad holds a slot array, which can be connected to each source pad.

3. CONCLUSIONS

The PadSpace architecture we have developed is a very extensible XML-based middleware. The Tuple Space computing is a very flexible architecture. There have been numerous extensions to the original Linda, and our work permits a tuple field to contain a new slot type, which plays the main role in connection interface (logically) between two or many composite pads. In our future work, we will also support Web services, with WSDL descriptions.

REFERENCES

[1] Y. Tanaka. Meme Media and Meme Market Architectures: Knowledge Media for Editing, Distributing, and Managing Intellectual Resources. IEEE Press, 2003

[2] Jun Fujima, Aran Lunzer, Kasper Hornbæk, Yuzuru Tanaka, Clip, connect, clone: combining application elements to build custom interfaces for information access, Proceedings of the 17th annual ACM symposium on User interface software and technology, October 24-27, 2004, Santa Fe, NM, USA