TOWARDS DISTRIBUTED OBJECT DESIGN

By

Ümit UZUN

1. ABSTRACT

Today's object-oriented (OO) software development methodologies use a number of graphical notations in analysis and design stages. There is an increasing research interest in incorporating distribution issues into these development methods. Allocation of objects and tasks, object replication and migration, remote interactions, multiple threads of control as well as network topologies are important issues in distributed object systems. In this report, ways to add distribution related issues to the object-oriented design methods (especially UML-based methods) are discussed.

We have implemented a case study in the area of Information Retrieval to concretize the issues. The implementation of the case study is done in Java [6] programming language, the source code of the programs is located at http://ftp.dcs.warwick.ac.uk/~umit/IR.

2. MOTIVATION ON THE CASE STUDY

Information retrieval systems are used to retrieve documents from a large number of document collections satisfying given criteria (queries). As an example, there is an information retrieval system under the search facility of a university library, where the types of documents are books, journals, theses, etc. Another example is the search engines on the Internet where the types of documents are web pages, articles in newsgroups, etc.

As the number of documents in the collection increases the process of executing the query and retrieving documents takes more time. The number of users making a query at the same time is also another issue; the performance of such a system decreases if more queries are being executed simultaneously. There have been various efforts to improve quality and performance of information retrieval. Using hardware with higher performance or using parallel computers is an effort in the hardware direction, so new hardware architectures are proposed for that purpose. Another issue is the use of different algorithms in the retrieval task: in the literature, there are quite a lot of algorithms developed to improve the search.

Distributed object computing may provide different solutions. Designing an information retrieval system using OO techniques, and implementing the system over a set of computers connected through a network may overcome this high performance computing need through using the resources already available and eliminating the need for new high performance hardware.

Implementing distributed object applications is now becoming easier using technologies like RMI [1] (Remote Method Invocation), CORBA [3] (Common Object Request Broker Architecture) or DCOM [2] (Distributed Component Object Model), however, there is not much research work on designing distributed object systems. The point of defining distribution concerns in the development methodologies is still unclear. In fact, the design of such an information retrieval system is going to be used to uncover those issues related to the design of distributed object applications.

The system is going to be designed in two stages. First, an object oriented design will be produced ignoring the underlying distribution platform and assuming the system will be running on one central machine. Next, a design including distribution aspects will be produced. The reason for doing so is to identify the differences between OO and distributed OO designs.

3. OO DESIGN OF AN IR SYSTEM

3.1 REQUIREMENTS

- The system is a multi-user information retrieval system, which is used to retrieve relevant documents to specific criteria from a collection of documents. The query will involve some search terms.
- The collection will include different type of documents, such as books, journals, on-line documents in different formats, etc.
- The system should allow introducing new types of documents as well, or the design should be easily extendible to include new types of documents in the collection. (Optional)
- Each type of document has its own properties; for example, a book has title, author and terms, etc., whereas an online document might have title, location, terms, etc., and an article may have title, author, abstract, journal, volume, etc.
- After the user inputs the query to the system, the system will retrieve a list of relevant documents.
- Users will be classified in terms of their rights, some users will be able to add/delete/update documents in the collection, some users not. Some documents will be marked as private and only users who have rights to access them will be able to retrieve them as a result to their queries (different access or operation policies might be defined).

3.2 CONSTRAINTS

- The number of documents and the number of users are expected to be huge. Therefore, one computer will not be sufficient considering the response time (performance) and space requirements, so there will be an underlying distributed computing platform. (a network of computers using TCP/IP as the network protocol)
- The system would be able to utilize different types of computer hardware and operating systems, such as Unix and Windows 95/NT.

3.3 NOTATION

UML [4] (Unified Modeling Language) notation is used in the design diagrams. The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting software systems, as well as for business modeling and other non-software systems. Extended information about UML notation is located in "UML v1.1 Notation Guide" on http://www.rational.com/uml.

3.4 DESIGN DIAGRAMS

Looking at the requirements above the most important use cases are: addition and deletion of documents and searching the document collection. The basic objects to be involved in such a system are: User, (Administrator), Document, Term, Document Collection, Search Engine and Query. The following diagrams explain the static structure of this system (relationship among objects) and the interaction among objects to make the system work.

3.4.1 USE CASE DIAGRAM

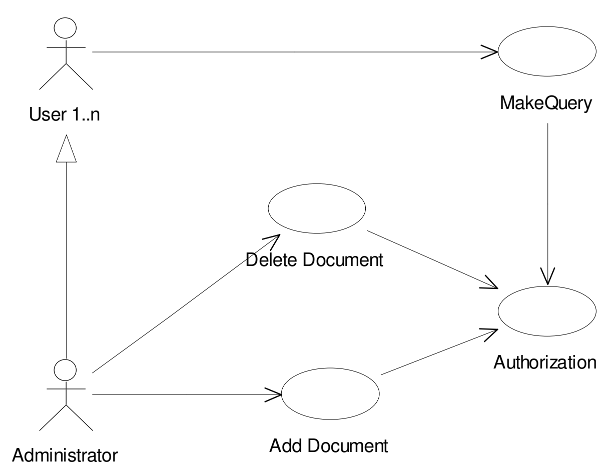

Figure 1 - Use Case Diagram

3.4.2 CLASS DIAGRAM

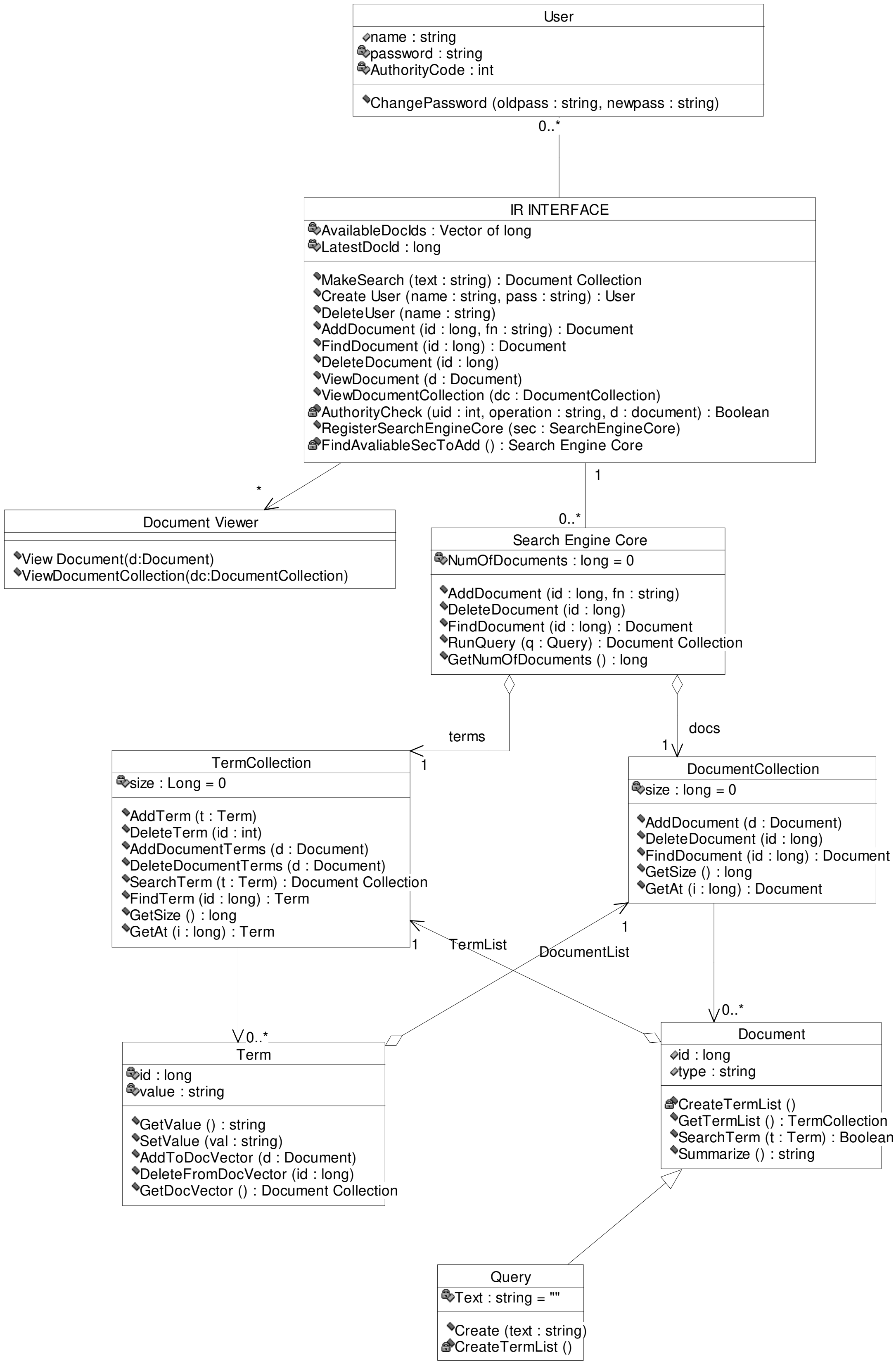

Figure 2 - Class Diagram

3.4.3 SEQUENCE DIAGRAMS

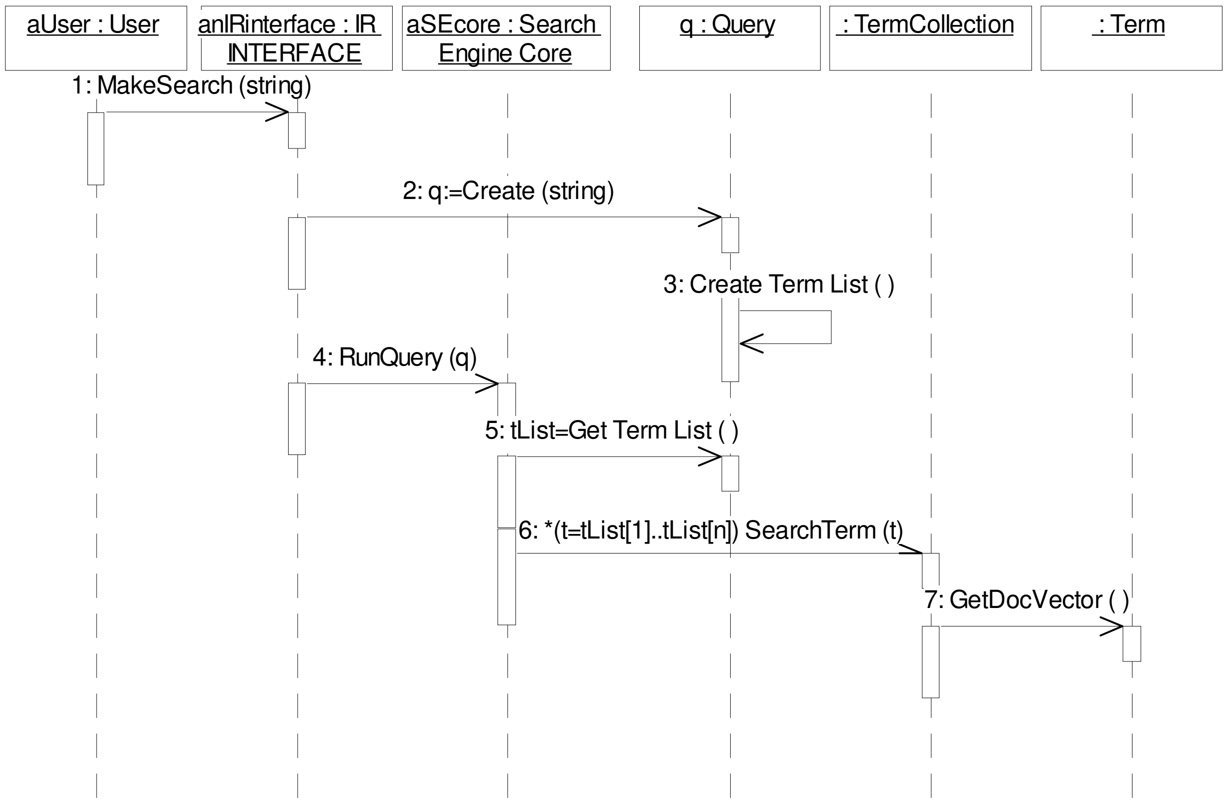

Figure 3 - Sequence Diagram for MakeQuery

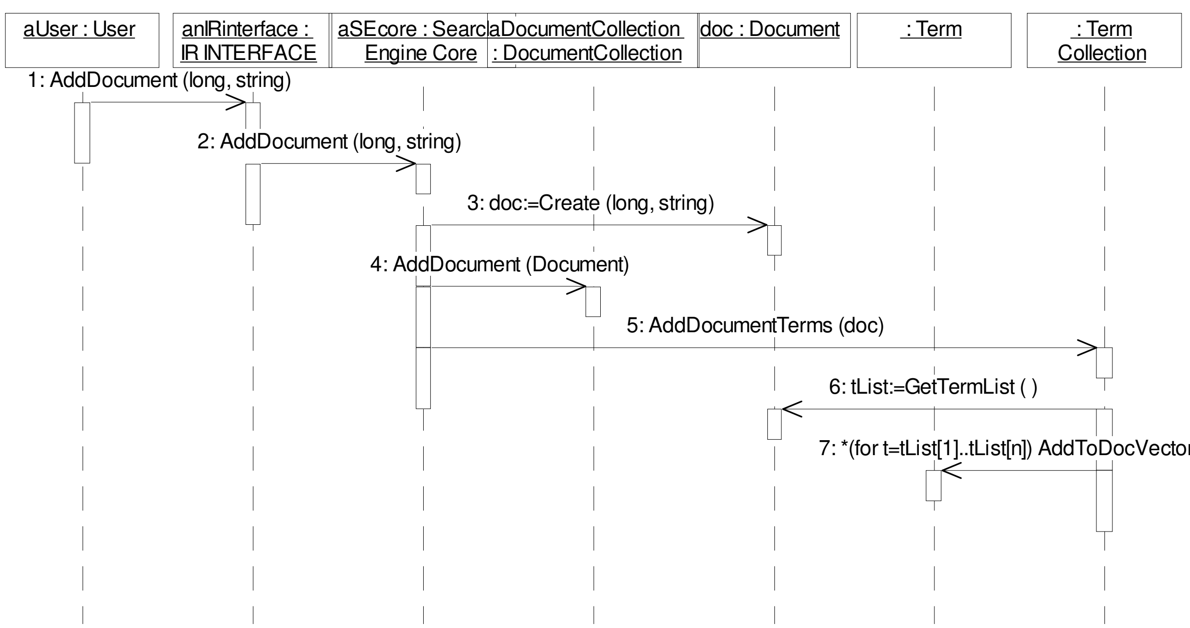

Figure 4 - Sequence Diagram for Add Document

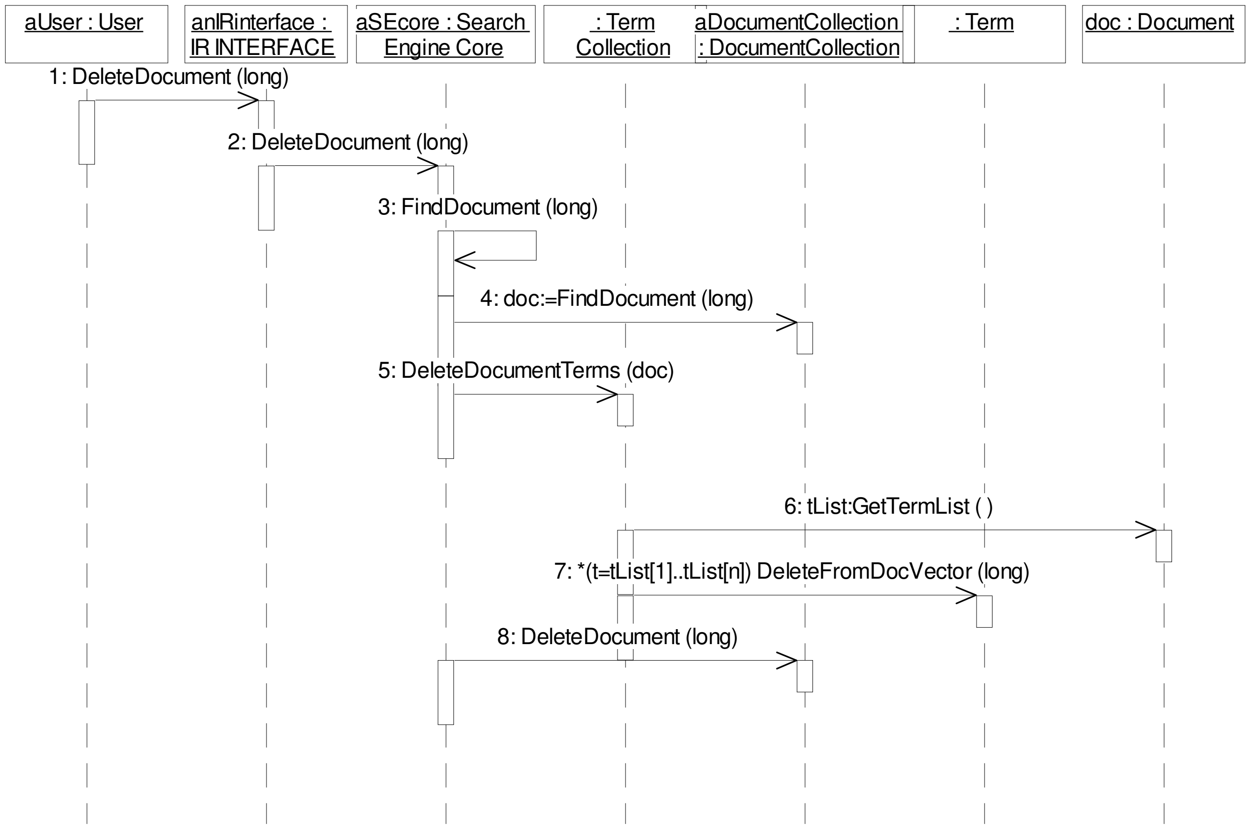

Figure 5 - Sequence Diagram for Delete Document

3.4.4 COLLABRATION DIAGRAMS

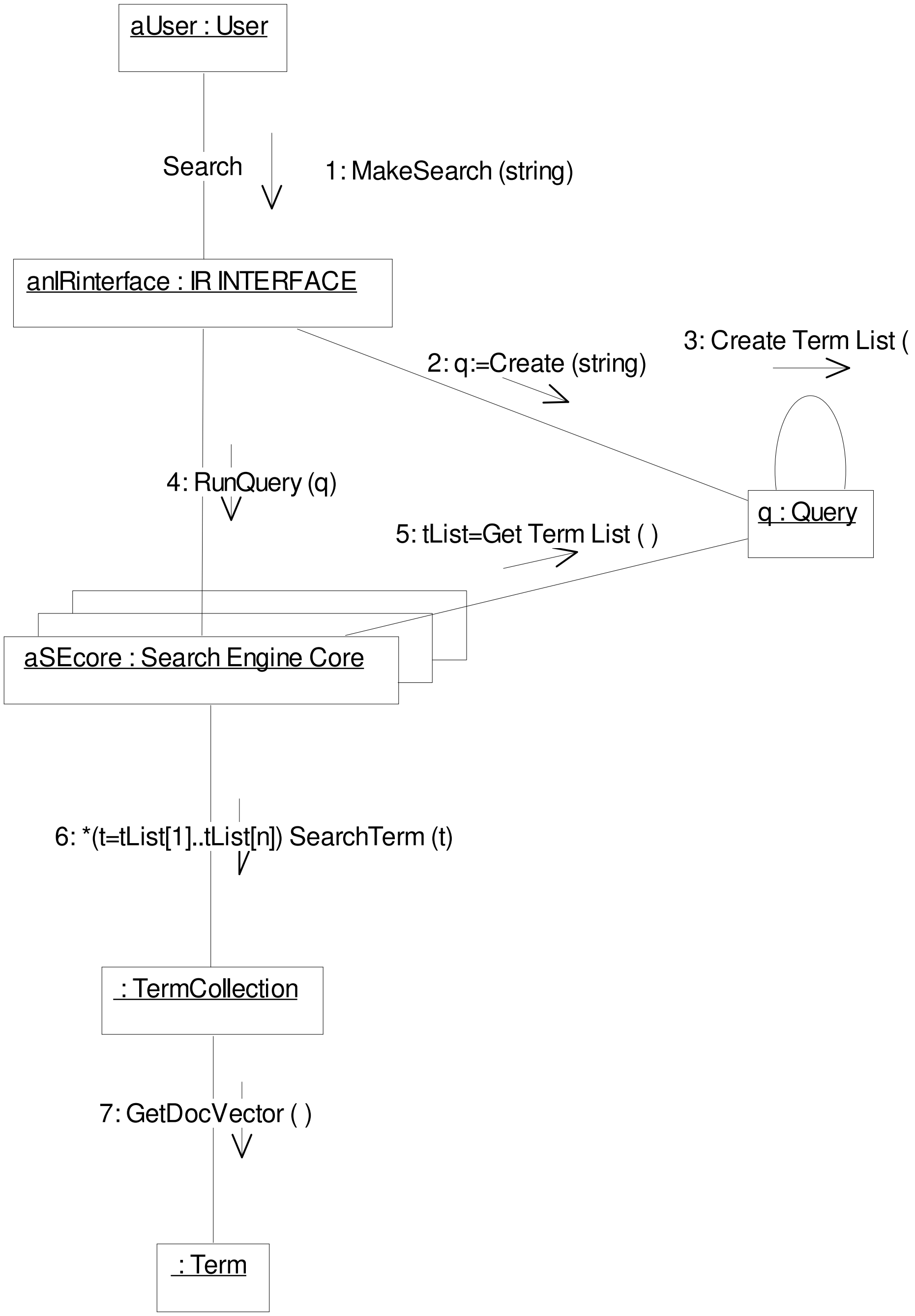

Figure 6 - Collaboration Diagram for Make Query

Figure 7 - Collaboration Diagram for AddDocument

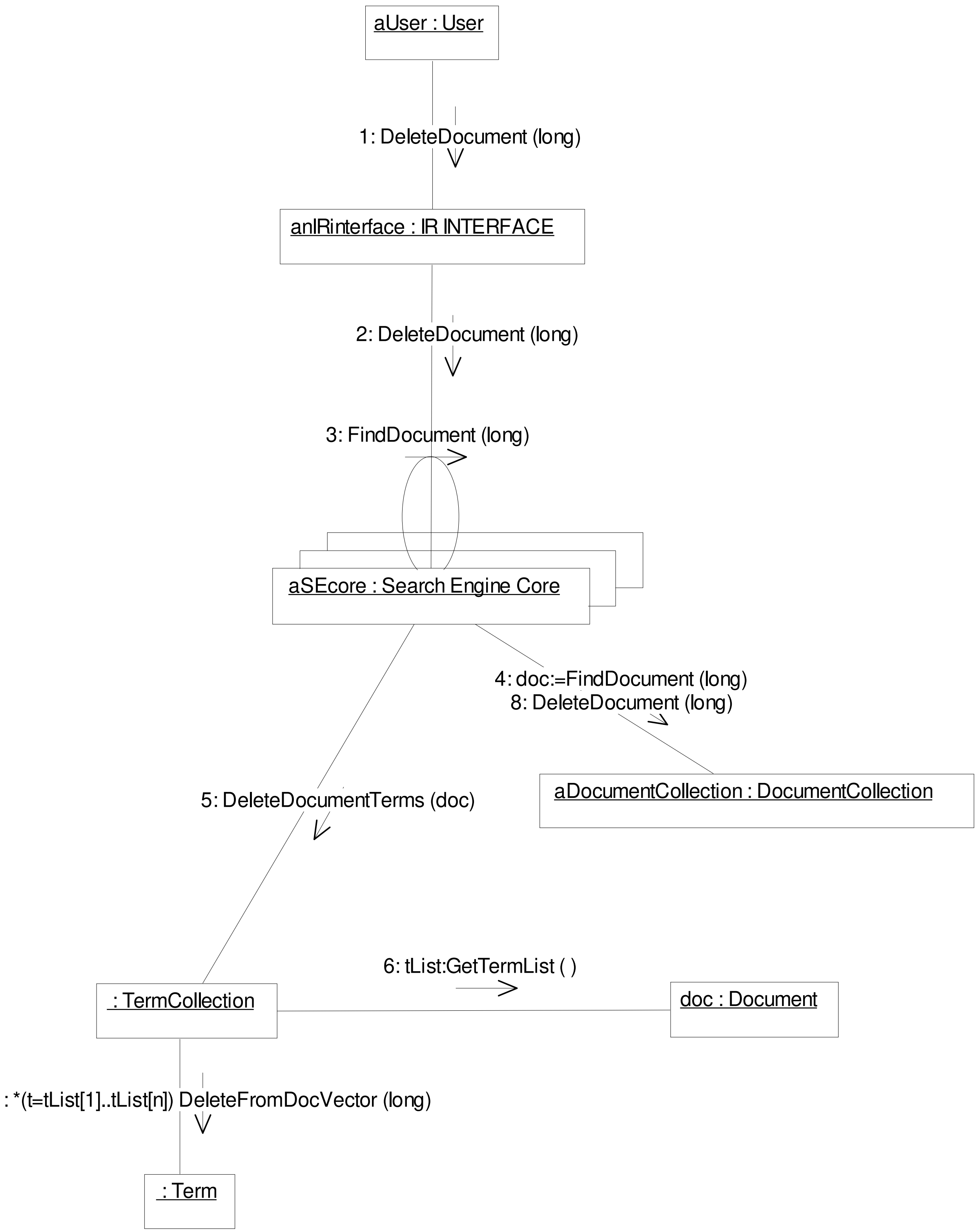

Figure 8 - Collaboration Diagram for Delete Document

4. ELEMENTS OF OBJECT ORIENTED DESIGN

Object oriented methodologies focus on determining the objects in a system, and their relationships and interaction. Several graphical notations are used to express the static structure and interaction of objects. The notation used in our case study is based on one of the most versatile methodologies, which is UML (the Unified Modeling Language).

In order to incorporate distribution issues in OO methodologies we should first look at the elements of object oriented design and attributes of those elements. Then, we will identify the new elements needed in the design for distributed systems. In the following parts elements and properties of those elements, which are most common in all OO methodologies, are shown. These fundamental elements of object orientation are discussed heavily in the literature hence detailed descriptions will not be included here. The following information will be useful to form a basis for distributed object design elements and properties.

Table 1 - STATIC STRUCTURE ELEMENTS

ELEMENT	PROPERTIES	Short Description
CLASS (OBJECT)		
	Name	The name of the class
	Attributes	Collection of *attribute*s belonging to the class
	Operations	Collection of *operation*s belonging to the class
ATTRIBUTE		
	Name	The name of the attribute
	Type	Type of the attribute
	Initial Value	Initial value of the attribute
	Access Control	Public, private or protected access to attribute
OPERATION (METHOD)		
	Name	The name of the method
	Return Type	Specifies the return type for the method
	Parameters	Collection of *parameter*s that is valid for the method
	Access Control	Public, private or protected access to method
	Preconditions	Specifies the predicates assumed by the operation (entry behavior of an operation)
	Postconditions	Specifies the predicates that are satisfied by the operation (exit behavior of the operation)
	Exceptions	Identifies the set of exceptions that can be raised by an operation
PARAMETER		
	Name	The name of the parameter
	Type	Indicates the data type of the parameter
	Call Semantic	Call by Value or Call by Reference
	InitValue	Sets the initial value of the parameter
ASSOCIATION		

	Name	The name of the association
	Roles	Roles incorporated in this association
ROLE		
	Name	The name of this role
	Class	Specifies a class belonging to the role
	Association	Specifies an association belonging to the role
	Cardinality	Determines the cardinality of the role
AGGREGATION (COMPOSITION)		
	Name	The name of the Aggregation relationship
	Client Class (this)	The client class of an aggregation
	Supplier Class	The supplier class of an aggregation
	Client Cardinality	The cardinality of the client class of an aggregation
	Supplier Cardinality	The cardinality of the supplier class of an aggregation
GENERALIZATION (INHERITANCE)		
	Name	The name of the Inheritance relationship
	Client Class (this)	The client class of an Inheritance
	Parent Class	The parent class of an Inheritance

Table 2 - INTERACTION STRUCTURE ELEMENTS

MESSAGE		
	Name	The name of the message
	Sender	The sender object of the message
	Receiver(s)	The receiver object (s) of the message
	Receiver Method	The method of the receiver object that responds to the message
INTERACTION		
	Name	The name of the Interaction sequence.
	Messages	The ordered collection of messages take place in the interaction sequence
STATE		
	Name	Name of the state
	Transitions	Collection of possible transitions to other states
	Nested States	Collection of states included in the state
EVENT		
	Name	Name of the event
	Transitions	Collection of transitions to trigger
TRANSITION		
	Name	Name of the transition
	Source State	The state which transition will occur from
	Destination State	The state which transition will occur to
	Destination Transitions	The transitions that will be triggered as destination (useful in forking)
	Guard Conditions	The conditions should be satisfied to trigger the transition

In the class diagram, Figure-2, we can see the static structure elements of a typical object-oriented design in a graphical notation, the following table includes examples from the class diagram.

Table 3 - EXAMPLES OF STATIC STRUCTURE ELEMENTS

CLASS (OBJECT)		
	Name	Document
	Attributes	Id
	Operations	Type
ATTRIBUTE		
	Name	Text (in Class Query)
	Type	String
	Initial Value	""
	Access Control	Private
OPERATION (METHOD)		
	Name	FindDocument (in Class DocumentCollection)
	Return Type	Document
	Parameters	Id
	Access Control	Public
	Preconditions	N/A
	Postconditions	N/A
	Exceptions	N/A
PARAMETER		
	Name	Id
	Type	Long
	Call Semantic	Call by Reference (Default)
	InitValue	N/A

In the Sequence diagram, Figure-3, we can see the interaction structure elements of an OO design in a graphical notation. Following table includes examples from the sequence diagram.

Table 4 - EXAMPLES OF INTERACTION STRUCTURE ELEMENTS

MESSAGE		
	Name	MakeSearch(String)
	Sender	aUser
	Receiver(s)	anIRinterface
	Receiver Method	MakeSearch(string)
INTERACTION		
	Name	MakeQuery
	Messages	1-anIRinterface.MakeSearch 2-Query.Create 3-Query.CreateTermList

		4-aSecore.RunQuery 5-Query.GetTermList 6-TermCollection.SearchTerm (*=repeated) 7-GetDocVector

5. DISTRIBUTED OBJECT COMMUNICATION

Distributed systems require that computations running in different address spaces, potentially on different hosts, be able to communicate. The primary goal of Distributed Object Computing is to provide solutions to allow communication between objects on different computing environments (usually connected through a network). Objects may reside on different types of architectures and might be implemented in different programming languages.

For a basic communication mechanism, there are sockets which can be seen as a communication channel between two networked computer (server and client), which are flexible and sufficient for general communication. However, sockets require the client and server to engage in applications-level protocols to encode and decode messages for exchange, and the design of such protocols is difficult and can be error-prone.

An alternative to sockets is Remote Procedure Call (RPC), which abstracts the communication interface to the level of a procedure call. Instead of working directly with sockets, the programmer calls remote procedures as if they were local, when in fact the arguments of the call are packaged up and sent to the remote target of the call. RPC systems encode arguments and return values. RPC, however, is not very suitable for distributed object systems, where communication between program-level objects residing in different address spaces is needed. In order to match the semantics of object invocation, distributed object systems require remote method calls (invocations). In such systems, a local surrogate (stub) object manages the invocation on a remote object.

The biggest effort to support the DOC idea is CORBA (Common Object Request Broker Architecture). After Java assumed a dominant role in Internet based programming, DOC became more important to the general users of OO techniques. Now all OO programming language tool providers are putting some level of remote object communication mechanism into their products. For example, Sun provides RMI for Java, Microsoft provides DCOM for Windows based programming, and the OMG (Object Management Group) defines CORBA for that purpose.

Amongst these CORBA is the most robust and provides the most complete solutions to the problems of DOC. Although RMI is only for Java, since the Java Virtual Machine (JVM) can run on heterogeneous platforms, it provides machine independence but not language independence. Microsoft's DCOM is providing solutions to a certain level under Windows platforms.

6. DISTRIBUTED OBJECT DESIGN

Object Oriented design methodologies are known to be a good way to develop applications. [5] As the communication technology improved, modern applications brought new needs in terms of making better use of new networking environments. New Object Oriented programming languages (Java) have arisen providing networking capabilities as language constructs. New specifications (CORBA) have been developed to implement distributed object computing, utilizing communication of remote objects living on different hosts. Consequently, most of the vendors in software industry have some sort of stake in Distributed Object Computing (DOC) nowadays. In DOC, the implementation level technologies are preceding design level methodologies, as happened for object orientation development itself.

We may define a distributed object oriented application as an object-oriented application where its objects are located in different hosts. Although this difference might not seem great, it requires many changes to distribute an OO application.

If we take the above case study as an example, the implementation of the system as a distributed application is not very difficult using RMI, CORBA, or other distributed object communication platforms. However, the design characteristics of such a distributed object system is not easily expressed in any of the current software development methodologies. Moreover, it is very difficult to predict the performance or error-prone parts of the system without having a good design prior to the implementation.

In this part of the document, necessary elements in distributed object design in addition to the elements of OO design will be discussed demonstrating problems that emerged from on the case study.

6.1 Problems arising in design of distributed object computing.

6.1.1 What to distribute? (Granularity)

The unit of distribution may depend on the application domain, problem size, possible parallelism (concurrency) potential in the system, physical constraints on resource allocation, etc. Coarser grain distributions will reduce the communication cost but decrease the level of distribution. On the other hand, finer grain distributions will increase the communication cost since even more messages will need to be sent across network, which is costly. The following questions related to our case study will make this problem clearer.

Shall we distribute the documents, terms, the whole search engine, collection type objects in the system or any object as small as an Integer?

Actually, all of these cases are possible. Intuitively (seeing a search engine as a worker in the system) we can say the best is to assign a different search engine to a different host (increasing the number of co-workers) , where each search engine is responsible for its own documents. In this case a central controller interface may control and/or query those distributed search engines. This is the basic supervisor/worker model of distributed computing.

Another option may be to distribute all the documents over the hosts and create many search engines, which are accessing the documents that they need. In this way, they could send broadcast messages to all documents hence increasing the concurrency. The same idea may be applied to any collections: distributing the items in a collection and accessing them concurrently.

One can think of different levels of granularity in distribution. The problem is which one is suitable or better with respect to certain criteria (the response time to a query in our case). Would it be possible to decide this without knowing the actual physical network characteristics? Would someone decide differently, if there were only a small number of hosts connected with a high-speed network or a large number of hosts with varying speed of connections?

We therefore need a way of specifying distribution units in our system and a way of evaluating the quality of this choice. None of the object oriented design methodologies supports this type of feature at this time.

6.1.2 Where to put? (Allocation)

Assuming we have identified the objects to be distributed, the question now is how to allocate objects to physical hosts. From our case study again shall we randomly distribute the objects over the network or would it be better in terms of performance if we allocate them in a predefined structure? Intuitively again we may argue that to put more objects onto more powerful hosts and put highly interacting objects onto closer (in terms of network speed) hosts would increase the performance. Is this true? If so, how are we going to prove that this is better than another way of allocation?

Another issue in allocation may come from the constraints in physical situations. For example, an object or the objects of a certain class may require a certain resource that is only available on a certain host, therefore this object should always reside on the host or hosts for which this resource is available. An object that prints to a printer may be this kind of an object where a printer may only be available only on some hosts

We can also consider the migration of objects in this section. Would the objects be able to move or would the reallocation of an object (migration) be possible in our system? If so, how is this going to be modeled in the design? The same argument applies to replication of objects. For reliability reasons, we may consider replicating objects.

Therefore, to handle the allocation problem we need a mechanism to specify and evaluate the allocation of objects in the object oriented design methodologies.

6.1.3 Physical / Virtual Network Schema?

As we have already mentioned, in order to answer some of the critical questions coming from distribution the physical network that a system will work on is also important. Therefore, we need to specify network characteristics in the design of the system, which should be integrated into the methodology used to specify the software structure of the system. Most of the methodologies leave this problem to the implementation level, claiming that the hardware to use wouldn't be a consideration in the design level.

However, in distributed object systems we believe that this is not true. We claim that the network characteristics play a significant role in important parameters (e.g. performance, reliability, cost etc.) of a distributed object system, hence the network to be used should be clearly specified and utilized in a distributed object design.

If we look at our case study again, without knowing or proposing a network structure we cannot say anything about performance or reliability of our design. For example, we would like to be able to say: with N hosts having a network with speed S and D documents the cost of a query operation would be C. Moreover, this information would be very useful in the case of a decision to upgrade. This way one could decide on the feasibility of adding a new host to the system or upgrading one or more of the hosts in the system.

6.1.4 How to cluster objects? (Clustering)

Clustering of highly interacting objects is an important issue. If we can analyze the message interactions between the objects to identify the highly interacting objects, and group them together it would greatly reduce the network traffic, hence increase the performance of the system.

UML and other design methodologies support collaboration (object interaction) diagrams, however the focus is only on the sequence of messages among objects. The objects are used as members of a certain class, therefore, collaboration diagrams are not enough to express exact interaction among objects. Another important issue is the multiplicity of objects and frequency of messages involved in an interaction, and this is also missing in the design. Differentiation of concurrent and sequential messages would also be useful.

This is also related to the allocation problem, because once a good way of clustering objects is found, the allocation of these clusters to different hosts would be useful. In our case study we don't know if we shall put the documents and it’s terms onto the same host or put the documents on one host and it's terms onto another one.

6.1.5 Exploiting and specifying parallelism.

One important use of distribution is making objects on different hosts execute concurrently. Therefore, we need to identify possible interactions that can take place at the same time. For example in the case of multiple queries, while one search engine is searching in a document another search engine may search in another document. Alternatively, in the case of single query, while one search engine is searching for a keyword another search engine may search for another keyword.

We therefore need to specify clearly which interactions are going to take place concurrently.

6.1.6 How to evaluate the quality of the design in terms of distribution?

Once the design of a system reaches a certain level one can begin worrying about the quality of the design in terms its expected performance on a distributed system. If there is any bottlenecks in the system, it is preferable to identify them in the design stage. If there were alternative ways of designing a part of the system, is it be possible to determine the better one in a rational or mathematical way?

6.1.7 Refinement of the design according to distribution issues.

As we can see, distributing an object-oriented system brings a whole lot of problems with its advantages, so the question of changing the design according to the distribution parameters is becoming inescapable. That change may cover addition of new objects (like proxies) or composition of some objects to a larger object to avoid high network interaction or decomposition of an object into smaller objects to increase parallelism.

However, without having necessary parameters about the design in terms of distribution, the revision of the design may become meaningless because the reason behind the change to the design would not be apparent.

We will discuss possible solutions to these problems in later sections.

6.2 Extra elements to be represented in a distributed object design

In addition to the elements listed above for the object oriented design, the following information should be represented at the design stage. So that evaluation of the design becomes possible and some of the ambiguities, which may cause a failure in the implementation of the system, may be resolved.

Node, Host

- Average memory available
- Average load

Communication Link

- Speed
- Average bandwidth available (or number of simple messages per time unit)

Remotely accessible object

-Location (specific node?)

-Services provided (interface)

-Mobility (allow to move?)

Message (method)

-Size (parameters and return value)

-Computation time

-Structure of message sequence in that message (including concurrency & syncronization)

-Number of target objects (1 or collection?)

-Frequency of this message to be sent

6.3 Proposed solutions

Firstly, the above elements should be included in the design. Some of those elements have already been introduced but others should also be included in a useful manner. Assuming we can have the above information from the design of the system, we can then begin answering the above questions.

6.3.1 What to distribute?

Introducing a notion of remote object will allow the designer to choose the level of granularity, i.e. once a class is defined to be remotely accessible, the objects of this class can be accessible from other hosts in the system. This way any class (basic or complex) may be marked as remotely accessible. Allocation of objects is another problem and will be explained later.

An example from the case study: if we want to make the Document objects remotely accessible then in the definition of the class we shall mark Document as a RA (Remotely Accessible) class.

6.3.2 Where to put? (Allocation)

Once we decide which classes are going to be remote then we can think of allocation of objects of this class. In other words, we shall decide the creation location of the objects in the first place. The choices are :

Minimum distribution: Force creation of all of the objects in a certain host.

Maximum distribution: Force the creation of objects in different hosts.

Optimum distribution: Force creation of objects on a lightly loaded host.

On the creator host: Create the object in the same host that creates that object.

Continuing the same example of documents, we may want to create every document on a specific host, or distribute them all over the hosts, or keep them on the same host where the search engine core is located that creates documents.

6.3.3 Physical / Virtual Network Schema?

As we discussed earlier, the description of the network is a necessity. We can use a descriptive language and/or graphical representation of the network.

Suppose we have 4 hosts in the system named varsity, gem, sun, mypc, we can roughly describe the network as follows:

Host:
- Name: Gem
- CPU: Sparc something
- Memory: don't know
- Average load: 50%
- Connected to: Ethernet Segment
- Connection speed: 100Mbps

Host:
- Name: Sun
- CPU: Sparc something
- Memory: don't know
- Average load: 20%
- Connected to: Ethernet Segment
- Connection speed: 10Mbps

Host:
- Name: Varsity
- CPU: Ultra 170
- Memory: 32M
- Average load: 60%
- Connected to: Ethernet Segment

Connection speed: 10Mbps

Host:

Name: MyPC
CPU: Pentium 200
Memory: 32M
Average load: 20%
Connected to: DialupNet
Connection speed: 33.6 Kbps

Network

Name: Ethernet Segment
Average bandwidth: 1Mbps

Network

Name: Dialup Net
Average bandwidth: 33.6 Kbps

Connection

End1: Ethernet Segment
End2: Dialup Net
Average Bandwidth: 5Mbps

Note that host descriptions implicitly define connections as well.

6.3.4 How to cluster objects? (Clustering)

In order to be able to do this we need to know a lot about the message interactions between objects. We may use a pseudo-code like language to define the message interactions only. This information can be obtained from collaboration diagrams as well. For the query operation, this code may be something like:

```
Operation
        Name: MakeSearch(str:String)
        Target Object: anIRInterface:IR_INTERFACE
        Input: String str  Size: 255;
        Interacting objects: q:Query(1);
                        tlist:TermCollection(1),
                        aSecore:SearchEngineCore(1),
                        dlist:DocumentCollection(1)
        Message Sequencing :
                q:=q.Create(str) . dlist:=aSecore.Runquery(q)
        Result: dlist
        RunTime: 10 units. (excluding messages sent to other objects)

Operation
        Name: RunQuery(q:Query)
        Target Object: anIRInterface:IR_INTERFACE
        Input: Query q,  Size: evaluated;
        Interacting objects: q:Query(1),
                        tlist:TermCollection(1),
                        t:Term(Many Avg:5)
                        dlist:DocumentCollection(1)
                        resultdlist:DocumentCollection(1)
                        d:Document
        Message Sequencing :
```

```
        tlist:=q.GetTermList() .
        for each t in tlist
            dlist:=t.GetDocVector()
            for each d in dlist
                resultdlist.AddDocument(d);
            endfor
        endfor
Result: resultdlist
RunTime : 0 (Excluding messages send to other objects)
```

Once we get the message interaction among objects, we may discover ways to find out which objects in the system are communicating more than others so that we can group them together. On the other hand, we have to relate this information to the allocation of objects since sending a message to an object on the same host and sending a message located on another host is different.

How to do the clustering is still a question but we believe clustering methods used in other areas of computing research (e.g. Information Retrieval) may be applied to here.

6.3.5 Exploiting and specifying parallelism.

This can be clearly specified in the message interaction scripts by introducing parallel activation of messages. For example, "||" notation can be used to specify two messages are going to be send at the same time and "par for" kind of notation can be used to specify parallel execution of loops. Thus for example

```
for each t in tlist
    dlist:=t.GetDocVector()
    for each d in dlist
        resultdlist.AddDocument(d);
    endfor
endfor
```

can be expressed as a parallel sending of messages as:

```
par for each t in tlist
    dlist:=t.GetDocVector()
    for each d in dlist
        resultdlist.AddDocument(d);
    endfor
endparfor
```

This sort of specification not only helps us identifying possible parallel sending of messages but also helps us determining possible problems in parallelism. In the above example dlist and resultdlist objects are being updated in a parallel execution which puts the system in a critical section therefore extra care must be taken in this case. A separate dlist object should be assigned for each t, and the update of resultdlist should be guarded.

6.3.6 How to evaluate the quality of the design in terms of distribution?

As we now have enough information about object location and object interactions we can calculate a cost function for each system-wide operation and object under certain configuration. Example cost functions that can be useful in our case study would be.

TimeCost of MakeSearch operation when Documents are randomly allocated to hosts.

TimeCost of MakeSearch operation when SearchEngineCores are randomly allocated to hosts and Documents are allocated with their SearchEngineCores.

From the above two costs if one is significantly smaller than the other one than we would make a rational decision on our design of distribution and be more sure that we are going in the right direction to implementation. Other criteria may also be important such as memory and CPU usage of different hosts under a certain input. In order to be able to evaluate these costs we should integrate the logical design and physical design characteristics of the system. There are two ways of evaluating the design, by calculation or by simulation.

By Calculation :

We need to extract required values from the design and express them in a manner that we can use in a calculation. We can set up different tables to achieve this such as :

Host Description Table(System wide)
Host-Conection Table (System Wide)
Connection Description Table (System Wide)
Connection Connection Table (System Wide)
Class Description Table(System wide)
Object Allocation Table (System wide)
Object-Attribute Table (System Wide)
Object-Message Table (System wide)
Attribute Description Table(Object wide)
Message Description Table (Object Wide)
Message-Message Table (Message wide)

Having this information in terms of tables, we can calculate the required costs. However, our proposed method is going to be simulation.

By simulation :

In this method each entity (class, message, host, connection etc) in the design is going to be represented as an object and the message interaction is going to simulated as if it is actually running. But these objects are only going to change the cost that we calculate. For example to calculate the time cost of a MakeSearch operation the script in the definition of MakeSearch is going to be executed. The global time value is going to be initialized as 0 and memory and cpu usage of hosts and bandwidth of connections are going to be initialized to their default values. An example scenario is as follows.

```
Name: MakeSearch(str:String)
Target Object: anIRInterface:IR_INTERFACE
Input: String str  Size: 255;
Interacting objects: q:Query(1);
                     tlist:TermCollection(1),
                     aSecore:SearchEngineCore(1),
                     dlist:DocumentCollection(1)
Message Sequencing :
          q:=q.Create(str) . dlist:=aSecore.Runquery(q)
Result: dlist
RunTime: 10 units. (excluding messages sent to other objects)
```

Create an ie:IR_interface

Reduce memory on host H by size of(ie) (including references to other object

Create a connection to ie

Send Runquery(q) over connection

Call ie.MakeSearch()

Increase time by runtime

Increase load on host H

Create a Connection between ie and q

Send q.Create() over Connection

Connection will increase the time by calculating time using bandwidth and message size. If both objects are on the same host then time=0

Call q.Create()

This is going to execute Create's script which will use the conection to return the result.

Create a Connection between ie and aSecore

Send Runquery(q) over connection

Connection will increase the time by calculating time using bandwidth and message size. If both objects are on the same host then time=0

Call aSecore.Runquery()

This is going to execute Runquery's script which will use the conection to return the result.

Return dlist over connection

At the end of this simulation, we can obtain required information such as: the time it takes to handle the message, connections used, processing time on each host etc. Changing the allocation mechanism, we can then look at the same values and decide which option is better.

6.3.7 Refinement of the design according to distribution issues.

Since we can now have a metric for the distribution performance of our design we can make improvements in our design to resolve the possible bottleneck or optimization of some operations. The important issue is to prove that our changes in the design will yield a better design and we can show this by evaluating our design under certain criteria as we discussed earlier.

The ways of refining the design may be similar in different application domains, which may then form a design pattern for distributed object systems.

7. CONCLUSION

We believe that object oriented methodologies may play an important role in distributed systems. However, object oriented design methodologies do not have enough support for distribution issues. In this document we have introduced initial new elements that should be added into object oriented design techniques to address the distribution of objects. We also demonstrated the problems and possible solutions based on our experience on a case study.

8. References

[1] SunSoft, "Remote Method Invocation Specification", Sun Microsystems Inc., 2550 Garcia Avenue Mountain View, CA 94043, USA, 1997 http://www.javasoft.com/products/jdk/1.1/docs/guide/rmi/

[2] N.Brown and C.Kindel "Distributed Component Object Model Protocol - DCOM/1.0" Internet Draft, Microsot Corporation, Network Working Group, May 1996. http://www.microsoft.com/oledev/olecom/

[3] Object Management Group "Common Object Request Broker Architecture", OMG Technical Documentation Archive, 1997. http://www.omg.org/corba/

[4] G.Booch and J.Rumbaugh and I. Jacobson, "The Unified Modelling Language for Object Oriented Development Documentation Set", Version 1.1, Rational Software Corporation, September 1997. http://www.rational.com/uml/documentation.html

[5] Grady Booch. *Object-Oriented Analysis And Design With Applications*. The Benjamin/Cummings Publishing Company, Inc., 390. Bridge Parkway Redwood City, California 94065, USA, second edition, 1994.

[6] SunSoft, "The Java Language Specification", Sun Microsystems Inc., 2550 Garcia Avenue Mountain View, CA 94043, USA, 1995. http://www.javasoft.com/products/jdk/