Ontological Design Patterns for Distributed Frame Hierarchy

Dmitri Soshnikov

Department of Numerical Mathematics and Programming
Moscow Aviation Institute (State Technical University)
Volokolamskoye shosse, 4
Moscow, Russia
dmitri@soshnikov.com

Abstract

With a very important role played by ontologies in many aspects of modern IT, the issues of ontology engineering deserve special attention. One of effective representation architectures for ontologies with dynamic knowledge in distributed environments is distributed frame hierarchy [1]. In this paper we present typical ontological design patterns that appear in connection to that architecture. Those patterns describe knowledge distribution configurations that can be effectively represented by supported interoperability, and thus encompass the range of problems that can be effectively solved in the given framework.

1. Introduction

In the recent years we witness the growing tendency towards interoperability of different devices and platforms in the distributed environment, supported by the expanding infrastructure of global networks. In the field of AI, this interoperability is governed by the notion of **ontology** [2] as a mechanism to ensure coherent use of concepts in distributed knowledge-sharing environment. Ontologies therefore play a major role in the design of contemporary knowledge-based systems, because they provide initial (partly standardized) conceptualization upon which the additional domain knowledge can be structured [3]. The importance of knowledge-based technologies in structuring the global information space is further realized in the concept of Semantic Web [4] — the set of W3C standards for the new generation semantically rich ontology-based resource markup that will make the Web understandable not only by humans, but also by software agents.

Proceedings of the 5th International Workshop on Computer Science and Information Technologies CSIT'2003
Ufa, Russia, 2003

As we feel the need for creating large-scale repositories of knowledge, the issue of ontology design becomes more and more important. Lately, there is a tendency in both theoretical research [3] and in practical applications [5] to apply methods of traditional software design to **ontology engineering**, and in particular the notion of design patterns [6]. [3] shows similarities between traditional software design patterns and ontologies, while [5] introduces the notion of ontology design patterns with respect to designing ontology for a particular problem domain in Biology.

In this paper, we introduce the notion of **ontology design patterns** more broadly, and discuss a number of patterns targeted towards distributed knowledge sharing and reuse supported by the model of distributed frame hierarchy [1]. Those design patterns outline distribution configurations that are effectively supported by two types of interoperability provided, and thus they encompass the range of problems that can be effectively solved using proposed model. From this point of view design patterns augment the formal semantics presented in [7], which formally defines the range of possibly represented reasoning methods, by focusing on **best practices** that are used in real-life applications.

2. Distributed Frame Hierarchy Model

In the area of distributed reasoning and multi-agent systems there have been a lot of research on communication between intelligent systems, and standards have been developed (KQML, KIF, OKBC). However, in all the approaches agents have some internal reasoning

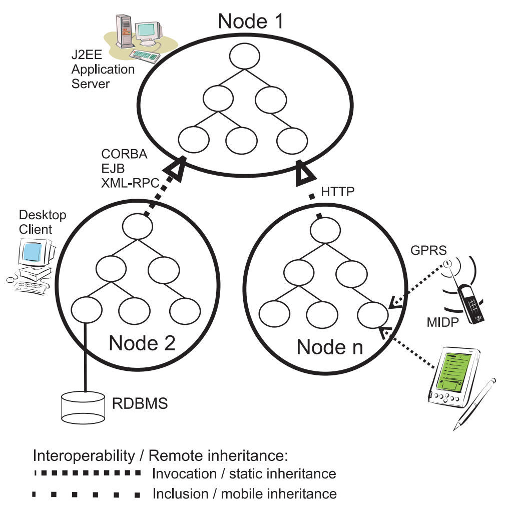
Node 1
J2EE Application Server
CORBA
EJB
XML-RPC
HTTP
Desktop Client
GPRS
MIDP
Node 2
Node n
RDBMS
Interoperability / Remote inheritance:
Invocation / static inheritance
Inclusion / mobile inheritance

Figure 1 Distributed Frame Hierarchy

mechanism and knowledge representation (which may be obtained by translation from the common ontology for multi-agent system), and they use some other knowledge representation for external communications.

While in many cases this seems like a very flexible solution, there are certain problems for which it seems to be too general and less efficient. In particular, we consider problems of *distributed knowledge sharing and reuse* for small and middle-scale applications, where we want to create a distributed knowledge repository and use it for performing reasoning over wide range of devices and platforms (for example, mobile terminals such as PDAa and cellular phones). For such systems we generally need lightweight implementation, and cannot afford overhead from translation between internal knowledge representation and external messaging protocol. In addition, we need lightweight and straightforward interoperability mechanisms that would consider network traffic minimization and work over existing protocols such as HTTP or IIOP/SOAP.

Thus, one of the goals is to unify knowledge representation throughout all nodes of the system, and also in inter-node communication. Since ontologies are normally organized as hierarchies of concepts with their properties in the form of declarative knowledge, we should look for knowledge representation that is structurally similar.

In fact, frame knowledge representation from classical AI is very much suited for this purpose, and it can be augmented by production rules for defining properties and statements about concepts (**production-frame knowledge representation**) [8]. As demonstrated in [1], this representation can be effectively distributed over computer network by placing different parts of the hierarchy on different nodes (Fig.1).

Each sub-hierarchy in such a model can be viewed as an independent hierarchy with the same knowledge representation and inference algorithms, having some external links to other hierarchies that we call **remote references**, i.e. references to frames and slots located in different subhierarchies. There are two types of remote references — **static** and **mobile** — that correspond to two types of interoperability: **remote invocation** and **inclusion**. Remote references are typically used in two contexts: in the `parent` slot for **remote inheritance**, or in any other context for **remote aggregation**.

The model of distributed frame hierarchy also provides seamless integration with relational databases and imperative software components (such as COM and CORBA-objects, JavaBeans, etc.). In the latter case the software object is exposed as a frame according to some naming convention, and relational table is represented by a series of pseudo-frames that reflect the data in the database.

In this chapter, only the most fundamental concepts of distributed frame hierarchy have been outlined. Other related publications (for example, [1]) provide more complete discussion of the subject, including such distinct features as **on-demand mobile referencing**, **constraints**, constraint-based and rule-based dynamic inheritance, etc.

Distributed hierarchy with inheritance that spans network node boundaries (**remote inheritance**) provides an effective framework for representing distributed ontologies augmented by dynamic knowledge from the problem domain in terms of production rules. Unlike the translation approach to ontologies [9], where ontological knowledge gets translated to the implementation language before it is used for logical reasoning, in distributed frame hierarchy the knowledge representation is auto-ontological, i.e. new pieces of knowledge get integrated into distributed ontology repository and are being available for immediate inference. However, the strict architecture of distributed frame hierarchy is more restrictive than generic multi-agent approach, thus it has some limitations and is most effective when utilised for a certain category of tasks that we refer to as **distributed knowledge sharing and reuse**. To outline the spectrum of possible applications more precisely and to highlight the best-use practices we define typical ontology design patterns.

3. Ontology Design Patterns for Distributed Frame Hierarchy

An architecture of distributed frame hierarchy presented above is not intended to be a general replacement for ontology design tools and approaches. Rather, it can provide more straightforward and lightweight solution to a series of typical problems of distributed knowledge sharing and reuse. Those problems are characterised by certain distribution of knowledge along different network nodes and the way this knowledge can be mapped to distributed production-frame knowledge representation.

By **ontological design pattern** we mean, adopting the definition from [6], a typical ontology configuration or knowledge distribution pattern that corresponds to effective and elegant solution of a certain problem. We present several knowledge distribution patterns that can be effectively represented using distributed frame hierarchy mechanisms. Where appropriate, an example of the design pattern is given, related to the medical problem domain.

Design patterns can be considered from two different viewpoints. First, they show which typical knowledge distribution scenarios can be effectively represented by distributed frame hierarchy, and how this can be done. They can also provide some recommendations on whether static or mobile inheritance/aggregation should be used. Secondly, if we provide fully comprehensive list of design patterns, we can argue that *only* their combinations can be represented effectively, thus defining more formally the range of application effectively handled by the proposed approach. While classification of patterns presented below is only the first attempt, and cannot be guaranteed to be comprehensive, we believe that the approach of characterising potential uses of distributed frame systems by corresponding ontological patterns is very constructive and convenient.

3.1. Notation

Fig.2 and 3 show schematic representation of the design patterns being discussed. On the pictures, circles correspond to frame subhierarchies, where filled circle denotes hierarchy with significant amount of knowledge included, while empty circle typically corresponds to simple hierarchy consisting of one or few frames with no production rules. Letters inside circles refer to problem domains (or subdomains) that knowledgebase is intended to. Arrows denote inheritance, aggregation (remote referencing) and other types of relationships according to the legend provided.

3.2. Inheritance Patterns

Most fundamental method for knowledge sharing in the distributed frame hierarchy is inheritance, which allows to apply knowledge stored in one subhierarchy to the remote problem data, to combine knowledge from two hierarchies, and to extend and augment knowledge contained in one hierarchy by rule sets located on different nodes. Still, usage of remote inheritance differs according to the exact purpose, which creates several inheritance-related design patterns outlined below.

3.2.1. Remote Consultation

Simplest form of inheritance is remote consultation (Fig.2A), where all knowledge is contained in the parent hierarchy, and a frame containing no rules is remotely inherited from it. By querying slots of this frame knowledge contained in the remote hierarchy is propagated through the network and applied, frame slots being filled by values derived in the process of inference. Depending on whether static of mobile inheritance is used, either slot values (static knowledge) are passed between network nodes in the form of remote calls, or the actual rules (dynamic knowledge) are exchanged. Former corresponds in a way to thin client application model, while the latter — to thick client[1].

A special feature to optimize network traffic in the remote consultation is **on-demand rule loading**, which is similar to mobile inheritance, but involves transferring only those dynamic rules that are actually needed during the consultation.

As an example, remote consultation can be used by a distant hospital to access central knowledgebase in the larger medical institution to perform a consultation, or by a doctor through his MIDP-enabled cellular phone.

3.2.2. Knowledge Reuse and Extension

In remote consultation, frame that extends parent hierarchy is used mainly for storing values (attributes) describing the problem being solved. However, in general it can also extend dynamic knowledge of the parent hierarchy with its own production rules, which may modify original knowledge to make it more accurate or suitable for solving particular problem. For instance, some specific cases may be identified, and for resolving all other cases knowledge from parent hierarchy can be used. This is the simplest form of knowledge reuse, when production rules from the parent hierarchy are used, but some functionality is extended (Fig.2B). This extension of the remote hierarchy need not necessarily be directly used for consultation, but it can also provide knowledge for further subclassing and reuse.

Considering the example above, a doctor in the distant hospital can extend the central knowledgebase with his own rules to reflect his experience. However, when the central knowledgebase is altered, this alteration would also be immediately available for the client.

Using static of mobile inheritance for knowledge reuse can be chosen depending on the specifics of the problem. A combination also could be used: for example, we can consider the doctor consulting central hospital server from a mobile phone using remote call, while the hospital server would inherit the basic knowledge from the variety of sources on the internet using inclusion.

[1]This terminology is not strictly correct, since in all cases software supporting operation of distributed frame hierarchy has to be installed on all nodes. Thin and thick clients in this case refer to where the inference takes place, and where the most of the knowledgebase (its dynamic part) is located.

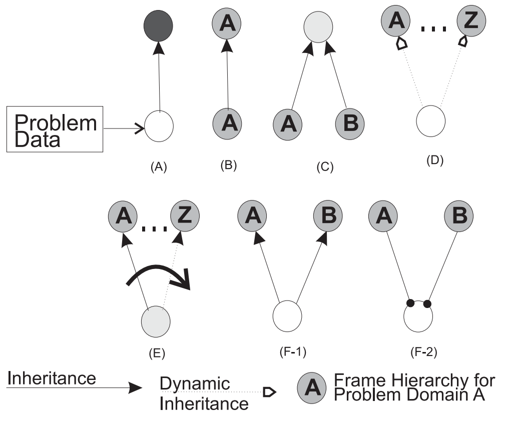

Figure 2 Ontological Design Patterns for Inheritance

3.2.3. Abstraction and Specialization

More common way of structuring knowledge in an ontology is **abstraction**, when some concepts are identified according to their base properties, and then they are extended to the number of more specific concepts, forming branch of the inheritance tree. Distributed frame hierarchy provides means for storing distributed ontological knowledge about concepts, and abstraction as a design pattern (see Fig.2C) plays here the same fundamental role.

Consider a central knowledgebase of a large hospital. It can be structured as an upper level ontology specifying such fundamental concepts as patient, department, treatment, etc., as well as some general diagnosing rules. Then, different departmental servers can extend this knowledge and adopt it to the specific problem subdomains.

3.2.4. Dynamic Inheritance / Subsumption for Problem Identification

One of the reasoning mechanisms in frame system is **subsumption**, when a concept is identified as belonging to a particular class. In the distributed frame hierarchy, one of the mechanisms of subsumption is dynamic inheritance, when a parent of a certain frame is determined either according to slot constraints, or by explicit production rules. This mechanism can also be identified as a design pattern (Fig.2D), when a separate frame is created, filled with initial data, and the process of subsumption inference is started to identify its parent. Identifying the parent of a frame in itself solves the problem of identification of the base class.

In the medical domain, we can have separate subhierarchies corresponding to different areas of medicine. Then, when a new patient is encountered, it can be classified to the correct subdomain by establishing inheritance link, after which deductive inference can take place to determine the exact diagnosis according to the given medical area.

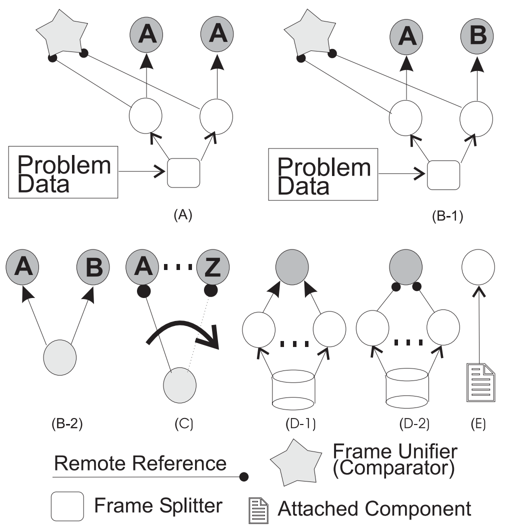

Figure 3 Ontological Design Patterns for Cooperative Problem Solving

3.2.5. Sequential Inheritance/Suitability Test

A process similar to dynamic inheritance is sequential (or pseudo-multiple) inheritance, when a frame is inherited from one parent at any given time, but then according to some meta-rules the parent is changed, making new knowledge available for application.

Sequential inheritance can be used for many purposes, for instance for filtering parent concepts according to a certain condition (suitability test), or as an alternative for dynamic inheritance. As an example of suitability test consider an expert system for choosing different domestic appliances from different vendors according to some criteria contained in *master frame*. Then, master frame can be sequentially inherited from the knowledgebases of different providers, which would determine whether their models are suitable for criteria, and suggest most appropriate model. After inheriting from all vendors in turn we would end up with a list of most suitable appliances.

3.2.6. Blackboard and Multiple Inheritance

Well-known from classical AI Blackboard architecture [10] can also be implemented via remote inheritance. The main idea of blackboard architecture is to have a common place for storing assertions about the problem being solved, which is accessible by a number of expert systems working cooperatively. In terms of distributed frame hierarchy one of the ways to achieve this is by multiple inheritance (Fig.2F-1) — however, in this case the knowledge from all parent subhierarchies would be applied in turn for solving the problem, and not cooperatively. Another possibility is remotely referencing the blackboard frame from several other subhierarchies (Fig.2F-2) — in this case slots of the blackboard frame can be cooperatively modified by asynchronous inference that takes place on the remote hierarchies.

3.3. Cooperative Problem Solving Patterns

Apart from inheritance, distributed frame hierarchy allows using remote references for any other purposes. This is mostly useful when there are different remote knowledgebases, and we want to use them in turn (or concurrently) to obtain the set of solutions to the same problem. This is typically referred to as **cooperative problem solving**, and has a number of associated design patterns.

3.3.1. Cooperative Problem Solving for One Problem Domain

Suppose there are different knowledgebases maintained distributedly by two specialists in the same problem domain. If they were developed using the same upper ontology, the attributes and naming conventions are likely to be the same. It makes sense to solve the problem using two knowledgebases on the same set of initial attributes, and then to compare the results. We refer to this pattern as cooperative problem solving for one problem domain (Fig.3A).

To perform such solution in distributed frame hierarchy, we need to create two identical low-level frames filled with initial data, which would also hold all intermediate reasoning results. This is achieved using the specific **frame splitter** component. When the inference is complete, results should be somehow combined, or the best solution selected according to some criteria. This is done by **frame unifier**, which can either be a specific component or normal frame referencing all solution frames, which determines the overall solution by some further inference.

3.3.2. Multi-Domain Problem Investigation

Another, in a way orthogonal, pattern of cooperative problem solving arises when problem domains are distinct. An example of this can be two different areas of medicine, and an attempt to diagnose certain set of symptoms from two different viewpoints. This can be accomplished by a pattern similar to previous one, only where problem subdomains of the parent subhierarchies are different (Fig.3B-1).

However, for different problem domains most domain-specific attributes are also likely to be distinct, or at least it is always possible to design an ontology with this goal in mind. In this case the corresponding pattern can be simpler (Fig.3B-2), and problem solution in multiple domains can be achieved just by (pseudo)-multiple inheritance.

3.3.3. Sequential Referencing

Two previous patterns of cooperative problem solving were designed to infer problem solutions independently, operating on separate base frames. However, sometimes we need to apply knowledge from several knowledgebases in turn to the solution of one problem, much in the same manner as in Blackboard architecture. However, while in Blackboard architecture inference may be asynchronous, here we want somehow to control the order of knowledge application.

In sequential referencing, we dedicate one frame as a base frame (or blackboard frame), and pass its reference along the series of frame subhierarchies responsible for maintaining actual knowledge. Those hierarchies would then modify some slots in the blackboard frame, thus making some steps towards the required solution, and then pass the reference to another subhierarchy to continue inference. It case static reference is passed — the actual blackboard frame resides on some active server, and blackboard values are read or assigned to via remote calls. For mobile references the actual contents of blackboard frame is passed from one network node to another, thus resembling the mobile agent approach, when the whole problem state is transferred over the network.

One of the practical applications of this approach is dynamic real-time search in the family of ontological knowledge-based active resource descriptions [11]. The idea behind it is to perform search in the family of resources annotated by knowledge-bases with common ontology by utilising an adopted graph search algorithm. The problem state in this case would be a queue of resources that have to be consulted and the result set, and by passing it from one node to another it would be enriched by further resources that are suggested by different knowledge-bases formed by intelligent resource descriptions.

3.4. Miscellaneous Patterns

3.4.1. Database Access

As it has been outlined above, a table or view in the relational database corresponds to a family of pseudo-frames, which typically reflects one of the following two design patterns:

- In **Frame-class inheritance** (Fig.3D-1) all frames in the family are inherited from a common parent that provides actual knowledge to be applied to the underlying data. This pattern is useful for filling in some fields in the table with reasoning results, or for querying certain inferred attributes of the underlying datasets.
- **Frame-class aggregation** (Fig.3D-2), when references to all pseudo-frames are used in some other manner, for example in the list-typed slot of a certain frame. This list can then be traversed during reasoning (using `MAP` operation), and results used elsewhere. This pattern is useful for filtering records according to certain criteria, or for computing some complex aggregate functions as the result of reasoning (eg., how many patients with coronary diseases are in a hospital – where a state

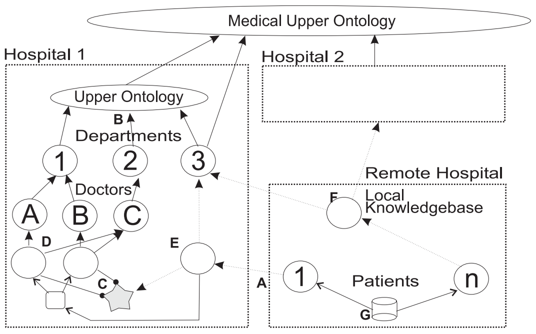

Figure 4 Combination of Design Patterns

of coronary disease is determined by reasoning according to the knowledgebase).

3.5. Imperative Components

Finally, we consider inclusion of imperative components in the form of pseudo-frames to be a design pattern as shown on Fig.3E.

3.6. Complex Example

The main idea behind ontological design patterns is that they can be combined to form more complex ontology architectures that can still be effectively implemented using distributed frame hierarchy. This idea is demonstrated on Fig.4, showing complex ontology architecture in the medical domain [12].

Taking a closer look at the picture one notices that it consists of individual design patterns that were given as an examples in the sections above. It includes remote inheritance for consultation (A), for knowledge abstraction (B), cooperative problem solving in one domain (C), and in orthogonal domains (D,E), combining knowledge from different sources (F), as well as database access (G). The diagram shown on Fig.4 can be considered as a graphical tool to model ontological interoperability within large distributed knowledge-based system; it can then be decomposed into more primitive design patterns, for which interoperability methods have been briefly outlined below. Thus. ontological design patterns can be considered as a graphical tool for designing ontologies based on distributed frame hierarchy suited for practical applications.

4. Conclusion

In this paper we have discussed the relation between ontology design and the architecture of distributed frame hierarchy, and have demonstrated that for problems of knowledge sharing and reuse distributed frame hierarchy can be used as a direct representation of domain ontologies. More formally, the domain of effectively handled problems have been characterized by a number of ontological design patterns, which, under all possible compositions, encompass the architectures of all effectively represented ontologies.

We believe that the approach of defining descriptive power of ontological design by design patterns is worth further investigation and formalization, because it provides constructive way to outline the set of all problems that can be effectively solved by a given underlying architecture. While strict semantic approach [7] defines the set of problems that can be solved, design patterns outline effectively representable structural and procedural knowledge configurations in terms of clarity and simplicity. Thus two presented approaches compliment each other, and contribute to the complete description of the given knowledge representation formalism.

References

1. Soshnikov D. An Architecture of Distributed Frame Hierarchy for Knowledge Sharing and Reuse in Computer Networks. In *Proc. of the 2002 IEEE Intl. Conf. on AI Systems*, IEEE Computer Society Press, 2002. – pp. 115-119.
2. Fikes R., Farquhar A. Distributed Repositiories of Highly Expressive Reusable Ontologies.*IEEE Intelligent Systems*, 3/4, 1999. pp. 73–79.
3. Devedžić V. Understanding Ontological Engineering, *Communications of the ACM*, Volume 45, No.4, April 2002, pp. 136-144.
4. Berners-Lee T., Hendler J., Lassila O. The Semantic Web. *Scientific American*, May 2001.
5. Reich J.R. Ontological Design Patterns for the Integration of Molecular Biological Information. In *Proc. of GCB-99 Conf. on Bioinformatics*, Hannover, Germany, 1999.
6. Gamma E., Helm R., Johnson R. and Vlissides J. Design Patterns - Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.
7. Soshnikov D. Interoperability Semantics in Distributed Frame Hierarchy. In *Proc. of* 4th *CSIT Workshop*, Patras, Greece, 2002.
8. Karp P.D. The Design Space of Frame Knowledge Representation Systems. SRI AI Center Technical Note #520, 1993.
9. Gruber T. R., A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2), 1993. – pp. 199-220.
10. Pfleger K., Hayes-Roth B., An Introduction to Blackboard-Style Systems Organization, *KSL Tech. Report KSL-98-03*, Stanford Univ., 1997.
11. Soshnikov D., Krasteleva I. ROMEO: An Ontology-Based Multi-Agent Architecture for Online Information Retrieval. In *Proc. of OntoBIS 2003 Conf.*, Colorado Springs, USA, 2003.
12. Lukianov I., Zavedeev I., Soshnikov D. Expert Ananysis in Chosing Medical Treatment Tactics for Patients with Infravesical Obstruction. In *Col. Abstr. of* 3rd *Conf. "Oncology Disease Treatment in Urology"*, Oncology Scientific Center, Moscow, 1999. — pp. 131–132. (In Russian)