Semantic Web: a Distributed Cognition view[1]

Sanjay Chandrasekharan

Center for Adaptive Behavior and Cognition,
Max Planck Institute for Human Development,
Lentzeallee 94, D-14195,
Berlin, Germany

schandra@mpib-berlin.mpg.de

Cognitive Science Ph.D. Program,
Institute of Interdisciplinary Studies,
2210, Dunton Tower,
Carleton University, Ottawa, Canada

schandr2@chat.carleton.ca

The World Wide Web is a complex socio-technical system, and can be understood in many ways. One dominant view looks at the Web as something like a library, where you search for and access information. In this view, the Web is a knowledge repository, albeit a very disorganized one, and the challenge is to get the maximum knowledge out of it in the minimum time possible. Most of the Semantic Web effort to date, and the work on search engines, assume this view of the Web.

However, there's another way of looking at the Web, which is to think of it as an *action-enabling-space*, where you can buy, sell, bid, book, gamble, play games, debate, chat etc. There is not much of an effort to understand and classify the Web from this point of view[2], as can be seen in the total lack of search engines that allow you to search for functions, like *sending_flowers*, *buying_tickets*, *booking_rooms* etc., though all these are activities possible over the Web.

The primary reason for this absence is the overarching nature of the first view – the web-as-information view – which subsumes the action-space view. This results in information about actions being treated as just another kind of information. So, if you need to buy tickets or book a room, you search Google; if you need to know more about buying

---

[1] Carleton University Cognitive Science Technical Report 2002-13
URL http://www.carleton.ca/iis/TechReports
© 2002 Sanjay Chandrasekharan
[2] The Web Services effort is a recent exception, but it caters more to programs than to humans.

tickets and booking rooms, you search Google as well, probably using the same keywords.

In this paper, we make a distinction between these two ways of understanding the Web, and argue that the design of the Semantic Web should focus more on actions possible on the Web, and develop ways to distinguish between search for actions and search for knowledge. In particular, we argue that the current design of top-down, exhaustive ontologies does not consider the representation of possible actions on the Web.

The following are the two major theoretical assumptions of this paper:

- The Semantic Web is a world-mediating system (Clark, 2001). According to Clark, "it mediates between users and a part of the world, often by manipulating machine representations of the world. State changes in the software system may cause state changes or side effects in the real world." In his article in *xml.com*, Clark explains this notion using the following example:

  "Consider a Web-based banking application. Performing banking tasks by using a Web application is functionally equivalent to performing them at the bank' s physical location. There are obvious phenomenological differences to the user in each case, but there aren' t any differences to the user' s bank account. A $100 withdrawal from a teller is equivalent, in all respects relevant to the bank account itself, to a $100 Web application withdrawal. A Web-based funds transfer just is a funds transfer, as a matter, among other things, of convention and institutional fact."

- From this view, of the Web as a world-mediating or action-mediating space, it follows that the development of the Semantic Web involves building *action-infrastructure* for agents, both human and artificial ones. That is, the design of the Semantic Web is akin to designing environments that support human actions in the world – environments like cockpits, kitchens and studios. The Semantic Web effort is thus about designing environment structures to fit the functions agents

want to perform in the world. The difference from cockpits and kitchens is that the actions performed on the Web are linguistic-acts, and therefore the environment designed to fit those actions is also a linguistic one.

Distributed Cognition and the Web

The view of the web as world-mediating or action-mediating turns the structure provided by the Semantic Web into affordances (Norman, 1998, Reed, 1996) for action. However, the commonly accepted view is that Semantic Web structures are knowledge representation structures, designed to facilitate knowledge recovery. So is the Semantic Web creating affordances or knowledge representation? Or both? Is there a distinction between the two? To find out, we have to first see how the Semantic Web can be considered as providing affordances for action.

Designing infrastructure for action is about tailoring cognition to the world. The building of such "congenial" structures for action in physical and representational environments has been explored in detail by the Distributed Cognition (DC) framework (Hutchins, 1995; Hollan et al, 2000; Kirsh 2001). Therefore, we can apply insights gained by Distributed Cognition to the design of the Semantic Web. However, since Distributed Cognition is not a design framework, but rather a framework for analysis, we have to develop a design framework based on Distributed Cognition to provide any design insights to the Semantic Web effort. We will sketch a skeletal design framework based on Distributed Cognition here.

DC is a theoretical framework that considers an agent's intelligence to be spread out among other agents[3] and functional contexts. Unlike situated cognition, Distributed Cognition recognizes the role of representation in action and cognition (For a systematic treatment of the distinctions between distributed and situated cognition, see Nardi, 1996, and Susi, 2001). DC considers cognitive processes as distributed across participants of a social group. The functioning of the cognitive system involves coordination between both

[3] Some theorists in DC focus just on the distribution of cognition across individuals and artifacts.

internal and external structures. DC also considers processes as distributed across time. This allows the framework to capture earlier events influencing later events. Unlike, say, task analysis, where the unit of analysis is the task to be performed and its complexity, the primary unit of analysis in the DC framework is the distributed socio-technical system, which consists of people working together and the artifacts they use. So the focus is on how people "bet on" structures in the environment (for another view on this, see Gigerenzer et al, 1999). In DC, individuals and artifacts are described as nodes, or agents, in the complex socio-cognitive system. Behavior is a result of the interaction between the external and internal representational structures.

The Distributed Cognition approach assumes that the analysis of a single individual's cognition in isolation will not provide us with an understanding of complex socio-cognitive systems like the Web. This is because the properties of complex cognitive systems, consisting of more than one individual, are not a sum of the cognitive systems of the participating individuals. Complex socio-technical systems have different cognitive properties from the cognitive properties of individuals that participate in such systems. For instance, take a collaborative task like programming. Programmers working together as a team will possess different kinds of knowledge. They will therefore engage in interactions that will allow them to pool the various resources to accomplish their task. These interactions can lead to results that are not a net sum of the knowledge the programmers have. Also, since knowledge is shared by the participants, communicative practices that exploit this shared knowledge can be used, like having a shared information structure, like a speed bug in a cockpit (Hutchins, 1995), or cryptic comments in code that exploit existing, shared, knowledge. Moreover, the distributed access of information in the system results in the coordination of expectations, and this becomes the basis of coordinated action.

Our analysis here will not follow the traditional Distributed Cognition methodology, which seeks to describe, through direct observation, how human agents create and interact with external structure and artifacts. The analysis here is more prescriptive, and will consider the role of the environment in different Agent Design frameworks and

suggest that one of them, where external structures are actively created for artificial agents, can be applied to the design of the Semantic Web.

The creation of structure in the environment for enabling action, or adapting the environment to the agent, has been explored within Distributed Cognition by Kirsh (1996), and to some extent Hutchins (1995). Kirsh's analysis considers how animals change their environment to make their tasks easier. He identifies two kinds of structure animals create in the environment, physical and informational. An example of physical structure Kirsh gives is the use of tools by animals, for instance Caledonian crows using twigs to probe out insects from the ground. The crows even redesign their tools, by making probes out of twigs bitten from living trees, and they fashion at least two different set of probes, one hook-shaped and the other pointed. Kirsh's example of informational structure created for action is people reorganizing their cards in a game of gin rummy. In this case, the player is using the cards to encode his plans externally. The cards "tell" the player what he needs to do, he does not have to remember it. The gin rummy algorithm is distributed across the player and the card set. The action of sorting the card set reorganizes the environment for "mental rather than physical savings". Kirsh (1994) terms these kind of actions "epistemic actions" as different from "pragmatic actions ". Epistemic action changes knowledge states, pragmatic action changes the state of the world. According to Kirsh, the second kind of structures created in the environment, informational structure, furthers "cognitive congeniality", as against physical congeniality, and is usually created only by higher animals.

We disagree with the second half of Kirsh's claim. We consider signaling, a very important aspect of animal life (cutting across biological niches) as an instance of changing the informational structure of the environment to further "cognitive congeniality". A simple thought experiment illustrates this. Consider the peacock's tail, the paradigmatic instance of an animal signal. The tail's function is to allow female peacocks (peahens) to make a mating judgment, by selecting the most-healthy male. The tail reliably describes the inner state of the peacock, that it is healthy (and therefore has good genes). The signal is reliable because it pays only a peacock with enough resources to produce a flamboyant tail. If you are a sickly male, you cannot spend resources to

produce ornaments. The health of the peacock is directly encoded in the tail; the peacock carries its internal attributes on its tail, so to speak.

To see the cognitive efficiency of this mechanism, imagine the peahen having to make a decision without the existence of such a direct and reliable signal. The peahen will need to have a knowledgebase of how the internal state, of health, can be inferred from behavioral and other cues. Let's say that "*good dancing*", "*lengthy chase of prey*", "*long flights*" (peacocks fly short distances), "*tough beak*" and "*good claws*" are cues for the health of a peacock. To arrive at a decision using these cues, first the peahen will need to "know" these cues, and that some combinations of them implies that the male is healthy.

Armed with this knowledge, the female has to sample males for an extended period of time, and go through a lengthy sorting process based on the cues (rank each male on each of these cues: good, bad, okay). Then it has to compare the different results, keeping all of them in memory, to arrive at an optimal mating decision. This is a computationally intensive process. The tail allows the female peacock to shortcut all this computation, and go directly to the most-healthy male in a lot. The self-description allows the peahen to have a single, chunked, cue, which it can compare with other similar ones to arrive at a decision. It provides a standardized way of arriving at a decision, with the least amount of computation. Reliable self-description, like the peacock's tail, is one of nature's ways of avoiding long-winded sorting and inference. In creating the Semantic Web using self-descriptive structures, we are seeking to emulate nature's design.

Note that the signal provides cognitive congeniality to the receiver, and not to the sender. The sender, the peacock, gains because he has an interest in being selected for mating. Kirsh's analysis considers how individual organisms change the environment for *their own* cognitive congeniality, and his claim about higher animals is probably justified in that context, because most animals do not create information structures for reducing their own cognitive complexity.

However, the reduction of others' cognitive complexity using signals is so common that it can be considered one of the building blocks of nature. Signaling exists at all levels of

nature, from single celled bacteria to plants, crickets, gazelles and humans. Surprisingly, this basic structure of cognition, where the information structure of the environment is changed to facilitate later iterations of a task, has received very little attention from Artificial Intelligence (See Hammond et al, 1995 for an exception). Many papers have considered the role of stigmergy in changing the environment structure. Stigmergy is a coordination mechanism where the action of one individual in a colony triggers the next action by others (Susi, 2001). It is a form of indirect communication, and has been a favoured mechanism for situated AI because it avoids the creation of explicit representations. Signaling, on the other hand, is closer to being a representation, and therefore more useful in understanding the creation of representations, like in the case of socio-technical systems like the Web. In the following section we look at how creation of information structures like signals can be incorporated into Agent Design. We do not subscribe to strong versions of AI, and just consider agent architectures as design methodologies here, after Bryson (2000).

Agent Design based on DC

We categorize Agent Design into four frameworks. We illustrate these four frameworks using the problem of giving physically handicapped people access to buildings. There are four general ways of solving this problem.

- **Approach I**: The first one involves not incorporating detailed environment structure into the design, and building an all-powerful vehicle, which can fly, climb stairs, detect curbs etc.

- **Approach II**: The second one involves studying the environment carefully and using that information to build the vehicle. For instance, the vehicle will take into account the existence of curbs, small stairs and elevators, so it will have the capacity to raise itself to the curb, a couple of stairs, or into an elevator.

- **Approach III**: The third one involves changing the environment. For instance, building ramps and special doors so that a simple vehicle can have maximum access. This is the most elegant solution, and the most widely used one.

- **Approach IV**: The fourth one is similar to the first one, but here the environment is all-powerful instead of the vehicle. The environment becomes "smart", and the building detects all physically handicapped people, and glides a ramp down to them, or lifts them up etc.

The first approach is similar to the traditional AI one (see footnote 1), which ignores the structure provided by specific environments during design, and tries to load every possible environment on to the agent, as centrally stored representations. The agent tries to map the encountered world on to this internal template structure.

The second approach is similar to the situated AI model promoted by Rodney Brooks (1991)[4], which recognizes the role of the environment, and analyses and exploits the detailed structure that exists in the environment while building the agent. Notice that the environment is not changed here. This is a passive design approach, where the environment is considered a given.

In the third approach, the designer actively intervenes in the environment and gives structure to it, so that the agent can function better. This is Active Design, or agent-environment co-design. The idea is to split the intelligence load — part to the agent, part to the world. This is agent design guided by the principle of Distributed Cognition, where

[4] Brooks questioned the traditional picture of Artificial Intelligence, where a centrally stored, idealised, representation of the world is stored within the agent and compared with the environment. The program executes actions based on these comparisons. Brooks observes that to port to a system, this notion of intelligence needs an objective world model provided by the programmer. This model would then be compared against "situations" and agents in the world. As Brooks has convincingly argued, this is not a robust way of building intelligence, because the world does not always fit the models made by the programmer. Instead, Brooks advocates a design where the designer considers the environment's structure in detail and builds low-level perception-action pairs (like *obstacle-run_away*) based on that structure. The agent constantly queries the environment to gain information on the structure of the environment, and acts on the basis of that information. This is a more robust design. Unfortunately, in the process of developing this design framework, Brooks took a stance against representations, which has resulted in this design framework not being applied much in representational domains like the Web.

part of the computation is hived off to the world. Kirsh (1996) terms this kind of "using the world to compute" *Active Redesign*.

This design principle underlies many techniques to minimize complexity. At Kirsh's physical level, the Active Design principle can be found in the building of roads for wheeled vehicles. Without roads, the vehicles will have a hard time, or all vehicles will need to have tank wheels. With roads the movement is a lot easier for average vehicles. This principle is also at work in the "intelligent use of space" where people organize objects around them in a way that helps them to execute their functions (Kirsh, 1995). Kitchens and personal libraries are instances of this.

A good example at Kirsh's information level (the cognitive congeniality level) is bar coding. Without bar coding, the checkout machine in your neighborhood supermarket would have to resort to a phenomenal amount of querying and object-recognition routines to identify a product. With bar coding, it becomes a simple affair. The Semantic Web enterprise is another instance of Active Design at the information level[5]. The effort is to create structure in an information environment (the Web) so that software and human agents can function effectively in it. At what we consider as the physical level of the Semantic Web, the Active Design principle is also at work in the Auto-ID[6] and the Physical Markup Language efforts, where products are provided with Radio-frequency Identification (RFID) tags containing information, which can be detected by RFID readers. Such tagged objects can be easily recognized by agents fitted with RFID readers, like robots in a recycling plant. The tags essentially create a referable world for such agents (See Chandrasekharan and Esfandiari, 2000, for more on the relation between agents and worlds).

---

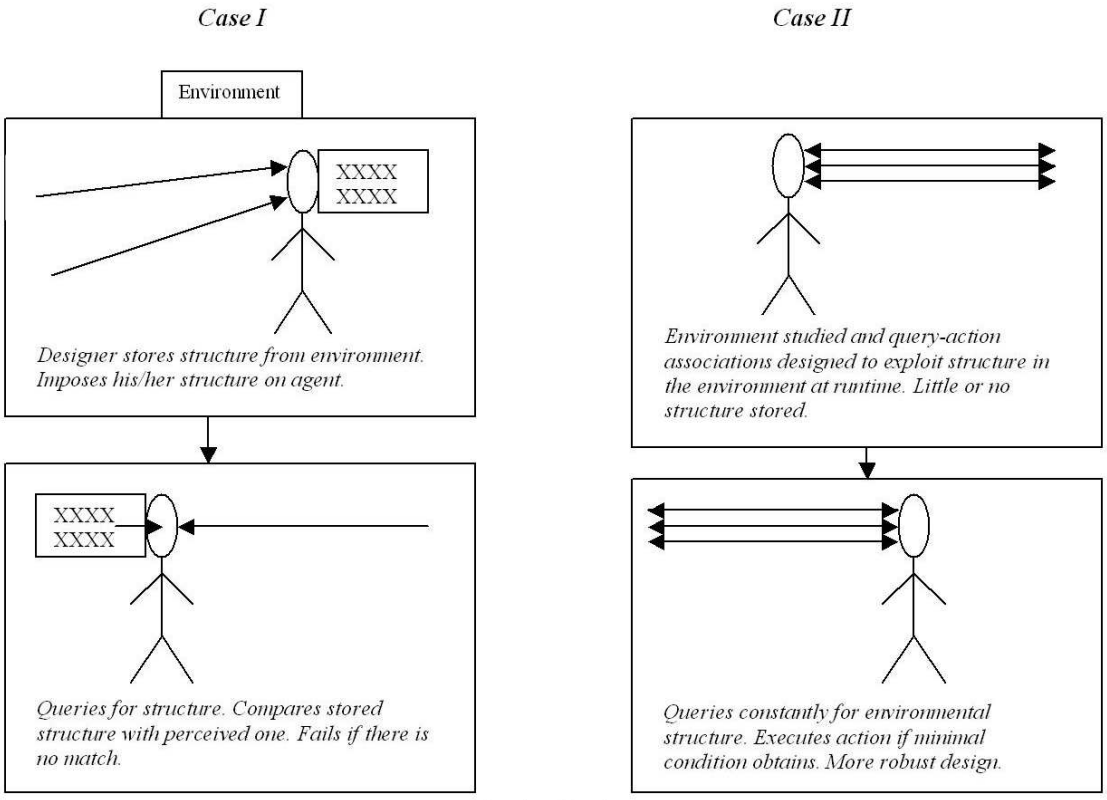
[5] Interestingly, if we consider the Semantic Web effort as the most recent development in the history of processing natural language, it follows the three design levels outlined above. First, in the era of NLP, language was considered as something that could be processed by using centrally stored rules and representations (first design approach). Then came automatic classification, the idea of trying to understand the structure of a document based on its context and domain, using pattern analysis and vocabularies (second approach). And now we have the Semantic Web, where the designer actively seeks to change the document environment, by providing structure to the document.

[6] http://www.autoidcenter.org/main.asp

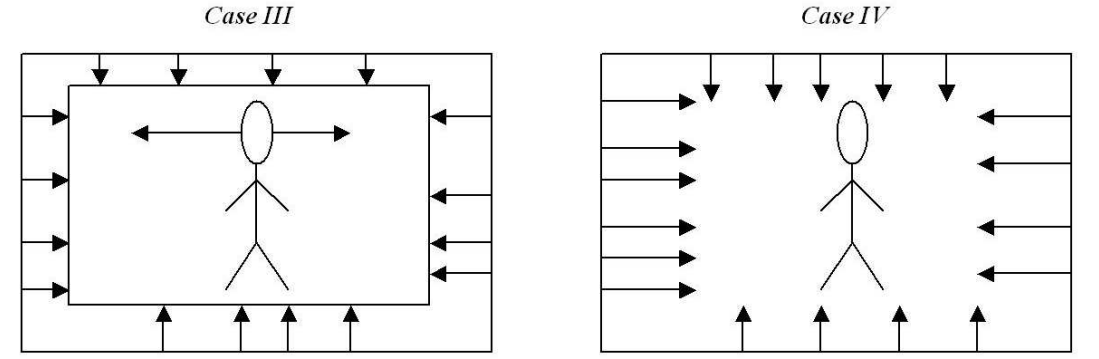
The Active Design approach is applied at the social level as well, especially in instances involving Trust. Humans actively create structure in the environment to help others make trust decisions. Formal structure created for trust includes credit ratings, identities, uniforms, badges, degrees, etc. These structures serve as reliable signals for people to make trust decisions. Less reliable, and more informal, structure we create include standardized ways of dressing, talking etc.

The fourth approach in our agent design taxonomy is the ubiquitous/pervasive computing idea. This is an extreme version of the Active Design approach.

Real design can be seen as a combination of two or more of these approaches. As illustrated by the examples, the third approach is the most elegant one — change the world, redesign it, so that a minimally complex agent can work effectively in that world. The picture below tries to capture these four design approaches.



Passive Design



Active Design

The four approaches to agent design are illustrated above. In the first two cases the environment is considered as a given, and the designer makes no changes to the environment. This is passive design. In the third case, the designer actively intervenes in the environment and gives structure to it, so that the agent can function better in it. The agent only queries for the structure provided by the designer. This is active design, or agent -environment co-design, where the knowledge is split equally between the agent and the environment. The agent and the environment evolve together. In the fourth case, it is the environment that is designed, and the agent is assumed to have minimal capabilities.

Semantic Web and Active Design

We suggest Active Design as a skeletal design framework built on the Distributed Cognition principle, where there is an active effort to split the cognitive load, and make the environment work for the agent. We claim that the Semantic Web effort is an instance of Active Design, because the Semantic Web effort seeks to provide structure to documents and programs (the environment in this case) so that other programs (software agents) and people can interact with them better. The structure provided "stabilizes" the environment (Hammond et al, 1995) for particular functions other agents want to execute. This way of looking at meta-tags – as created structure for agents' functions – puts meta-tags closer to the ecological psychology notion of built environment and created affordances (Reed, 1996) than to knowledge representation.

All standardizations can be considered as building such stable environments for actions, for instance library classification schemes stabilize document environments for search. The stabilization can be at different levels, and the different metadata formats create different structure. Thus XML provided standardized syntax and RDF provides a standardized description of resources. Notice that while low-level standardization (like processor speeds and XML) provides stabilization that supports a variety of functions, high-level standardization (like library organizations and filename extensions) usually are designed to "fit" particular functions, like high -level search, instead of cue-based search.

There are quite a few advantages to distributing structure out into the environment. In human agents, the commonly cited functions of externalized representations are the reduction in the load for memory, perception and attention (Kirsh, 1995b). Cox (1999), who works in the problem-solving paradigm of external representations, provides a list of how actively created external graphic representations assist in problem solving. The following is a non-exhaustive list of advantages externalized structure provides for human agents:

- Translating information from one type of representation to another
- Re-ordering information in useful ways

- Directing attention to unsolved parts of the problem
- Organizing information spatially
- Keeping track of progress through the problem (allows for breaks, multitasking etc.)
- Transferring information between cognitive subsystems
- Changing what is recalled
- Facilitating the inference of motion (mental animation)
- Shifting the subject's mode of reasoning

Many of these advantages are applicable to artificial agents as well. For instance, the translation to other representational formats and transfer of information between subsystems is a very clear advantage provided by an externalized representation in the case of multi-agent systems. In general, we can say that the primary reason we change the environment is to keep the agent simple, to reduce complexity for/of the agent.

How does the change in the environment reduce complexity? The complexity-reduction happens through the focusing of structures to function (See Agre and Horswill, 1997 for an elaboration of this point). Essentially, the created structure in the environment fits the function the agent seeks to perform. A good example from the animal world is again the tail of the peacock, which fits the mate selection function of the peahen. An artifact example is bar coding. The information in the barcode is extremely focused to the functioning of the supermarket check-out machine. It provides the machine information like the name of the product, weight, date of expiry and price. It does not say that the container is round, or is made of plastic or that it was made during foreman Bill's shift at a factory in Paradise Falls. The bar code could be easily made to say all that, and there could be functions that need such information. For instance, one can imagine a barcode which says that "this is made of plastic", and a machine in a recycling plant using that barcode to sort plastic containers.

Notice that the check-out machine has no use for this information, and that the presence of this information only adds complexity to the check-out machine. The optimal design solution is where the agent (the check-out machine) gets *just* the information it needs,

like price etc. Similarly, the recycling machine needs only information on the type of material, not the prices and date of expiry and whose shift the container was made. For Active Design to work best, the structure provided to the environment should focus on the *function an agent needs to perform*. A uniform, generalised, structure that is potentially useful for all agents is not an efficient structure. Such a generalised structure would only add complexity to the processing any single agent needs to perform, because such a structure would increase search, as it is not focused to the function the agent wants to perform.

Ontologies as affordance

Let's apply the above insight to the Semantic Web, or to one aspect of it, namely formal ontologies. A formal ontology is a specification of a conceptualization, or in less formal talk, a standardised representation of concepts and their relations. An Ontology metatag essentially classifies a document as being part of a knowledge domain (or a real world domain). So if you have a document with the metatag ***cs:***, that means the document is about a computer science department. Now let us say you have an agent that roams the web and collects the names and e-mail addresses of all the faculty members in computer science departments. Once the ontology part of the Semantic Web is in place, all the agent needs to do is look for the ontology meta-tag on a web page, and then parse the document to get the name and e-mail of the faculty member. Notice that the rest of the information in the page is irrelevant to the function the agent wants to perform. Also notice that the harvesting of professors' e-mail is not a function that the developer of the ontology necessarily wanted to support[7]. However, if the professors' pages refer to an ontology that provides just enough information to support particular functions, this kind of exploitation of structure would be more difficult.

Now, consider another example, this time from the Physical Markup Language (PML) domain. PML allows us to provide structure to everyday objects using RFID tags.

[7] In animal communication, this kind of undesirable exploitation of informational structures by others is termed "eavesdropping". For example, the songs male crickets sing to attract females are used by some parasitic flies to locate the male crickets and deposit their eggs on them. The flies kill the cricket when the eggs hatch.

Suppose that we want to mark up a coffee cup so that a housemaid robot can find it and bring us coffee. We can markup the cup in, say, the following manner:

```
[Object]
   Container
      Cup
       Coffee cup
            Jim’s Coffee cup
[Measure-ont]
  LengthUnit 20 CM
  WeightUnit 200 Grams
  VolumeUnit  77.9 CM^3
```

This markup in an RFID tag identifies the cup and provides some of its properties, and the robot can detect this cup using its RFID reader. Like the barcode, this information allows the robot to short cut object recognition routines. However, to execute its action of filling the cup with coffee, this information is still not enough. This is because nowhere in the markup does it say what are the functions the cup supports. To use the provided information to execute its function, the robot has to know that coffee cups are used for filling coffee, and the procedure to fill a cup with coffee is to hold it open-side up under the coffee machine’s tap after switching it on. It also has to know that it should not hold the cup upside down once the coffee is filled. Finally, the robot has to know what actions to select from its repertoire of actions to use on the cup to fetch coffee. A much more useful informational structure for the robot would be:

```
[object: Jim’s coffee cup; supported_functions: hold, fill;
constraints: this side up; properties: radius 3 cm, height
20 cm; volume: 77.9 cm^3]
```

Here the self-description provided by the cup explicitly tells the robot what functions it supports, and leads to a lot less inference by the robot. Of course, the cup can be used for a lot of other functions, like to measure rice, as a candle-holder, as a paperweight etc. But these are not the intended functions of a cup, and to put in all these functions in the tag

would make the structure-creation a never-ending exercise. It is better to put in the prototypic functions, and leave the rest to the creativity of the agents encountering the cup, as happens with humans. On the property side, the tag just includes the properties that are required for the agent to execute the functions suggested/desired. This makes the job of the tag designer (the equivalent of the ontology designer in the Physical Markup Language scenario) a lot easier, because function-based tagging means that a lot less information needs to be put in.

The basic issue here is the way the cognitive load is carved between the agent and the environment. Traditionally, functions have been considered as something the agent brings to the world (objects), and to execute its function the agent needed to know *just* the properties of the object, which are considered to be the object's *only* contribution to the decision. The agent can infer whether a given object supports the function based on the properties the object "possesses". In our approach, a part of the function can be in the object, as affordances for action. If the functions the agent wants to execute are the same ones the object "affords", there is a better "fit" between the world and the agent, and there is less cognitive overhead. If the structure and the action do not fit, the agent has to spend cognitive effort to fit the action to the object or vice versa.

From the point of view of making the environment work for individual agents, we consider referring to this kind of function-based ontologies by objects much more useful and efficient than referring to general purpose, exhaustive ontologies that require extensive search and inference on the part of the agent, because of the detailed categories and relations such ontologies contain. Functions in objects act as "lenses" that edit out unnecessary structure, both for agents and for tag designers. Also, functional ontologies help users, agents and search engines to easily discover web pages that provide functions like *buy, sell* and *bid,* allowing them to distinguish between the information part of the web from the functional part[8]. Functional ontologies will allow the web to be split into

[8]Even a successful web business like Amazon does not have meta-tags mentioning the functions they provide. Amazon's meta-tags are: *amazon.com,amazon books,amazon,amazon.com books,amazon music,amazon.com music,amazon video,amazon.com video,auctions,amazon auctions,amazon.com auctions,electronics,consumer electronics,gifts,amazon gifts,amazon.com gifts,cards,e-cards,e-mail*

action and knowledge domains, and allow for separate and detailed searches in both. Another advantage of putting functions in web pages is that we can create a network of functions, by linking pages by function, instead of knowledge. This is an instance of the point Cox makes, that having an externalized representation helps in the interaction between different cognitive systems. Another related advantage is the ability of agents to translate the function to different formats, which is not possible if the function just resides within the agent. Also, as pointed out earlier, by providing focused ontology snippets that support desirable functions, we could also potentially keep out "eavesdroppers" like e-mail harvesting bots.

However, this function-based approach to creation of structure does not contradict the design of top-down, exhaustive ontologies. This is because functional ontologies and exhaustive ontologies serve different purposes. The reason why we seek to develop top-down ontologies is the same as the development of any standard format: interoperability. In the case of ontologies, the standard format sought is a common vocabulary. The view of function acting as a lens (allowing agents to focus on only the needed environment structure for action) does not deny this standardization role of top-down ontologies. The functional view says just this: because we develop top-down ontologies to have a common vocabulary, the designer of an individual web page or an RFID tag should not have to put in (or refer to) an entire exhaustive ontology. The designer should be able to put in, or refer to, *ontology snippets,* focused to particular functions commonly served by an object. And she should be able to pick and choose snippets in any way she wants to, without restrictions based on hierarchy.

This means the answer to our question "is the semantic web about affordance or knowledge representation?" is: *it can be both*. The function-based approach seeks to exploit the existence of a common vocabulary, a knowledge representation framework, but it advocates the use of elements of that common vocabulary as affordances in objects, i.e. as action-structures focused to functions. This is similar to the way humans use ontologies – we always access only action-relevant parts of perceived objects, in relation

---

cards,greeting cards,amazon cards,amazon.com cards,toys,amazon toys,amazon.com toys,games,amazon games,amazon.com games,toys & games,toys and games

to task functions. We are selective about the parts of the environment we attend to during a task, and we almost never use all the properties of an object to execute a task. Thus the extension of this principle to the Web is quite natural if we take the stance of the Web as an action-mediating space. The inclusion of the functional structure as a "lens" in a self - describing object or website allows agents who need to perform that function to easily detect that structure, *and only that structure*, and use the structure to efficiently perform the task.

References

Agre, P. and Horswill, I.(1997) "Lifeworld Analysis." Journal of Artificial Intelligence Research, 6, 111-145.

Brooks, R. A. (1991). Intelligence without representation. Artificial Intelligence 47(1-3): 139 - 160.

Bryson, J. (2000). Cross-paradigm analysis of autonomous agent architecture, *Journal of Experimental and Theoretical Artificial Intelligence*, 12 (2), 165 – 189.

Chandrasekharan S. & Esfandiari, B. (2000). "Software Agents and Situatedness: Being Where?", in the Proceedings of the Eleventh Mid-west conference on Artificial Intelligence and Cognitive Science, Menlo Park, CA, AAAI Press.

Clark A. (1997). *Being There: putting brain, body, and world together again*, Cambridge, Mass., MIT Press.

Clark, G.K (2001) The Politics of Schemas, http://www.xml.com/pub/a/2001/01/31/politics.html

Cox, R. (1999) Representation construction, externalised cognition and individual differences. Learning and Instruction (Special issue on learning with interactive graphical systems), 9, 343-363.

Gershenfeld, N. (1999). *When things start to think*. New York: Henry Holt and Company.

Gigerenzer, G., Todd, P. M., & the ABC Group (1999). *Simple heuristics that make us smart*. New York: Oxford University Press.

Gruber, T.R. (1993) A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220.

Hammond, K.J., Converse, T. M., Grass, J W. (1995) The stabilization of environments. Artificial Intelligence, 72(1-2):305-327.

Hutchins, E. (1995). How a cockpit remembers its speeds. *Cognitive Science,* 19, 265-288.

Hollan, J.D., Hutchins, E.L., Kirsh, D. (forthcoming). Distributed cognition: A new theoretical foundation for human-computer interaction research. *ACM Transactions on Human-Computer Interaction,* in press.

Kirsh, D. (1990). When is information explicitly represented? In P. HANSON, Ed. *Information, language, and cognition*. (Volume I of The Vancouver Studies in Cognitive Science, 340-365) Vancouver, BC: University of British Columbia Press.

Kirsh, D., & Maglio, P. (1994). On distinguishing epistemic from pragmatic action. *Cognitive Science,* 18, 513-549.

Kirsh, D. (1995a). The Intelligent Use of Space. *Artificial Intelligence,* 73, 31-68

Kirsh D. (1995b). Complementary Strategies: Why we use our hands when we think. In Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum.

Kirsh, D. (1996). Adapting the Environment Instead of Oneself. *Adaptive Behavior*, 4 (3/4), 415-452.

Kirsh, D. (1999). Distributed Cognition, Coordination and Environment Design. *Proceedings of the European conference on Cognitive Science.*

Kirsh, D. (2001). The Context of Work, *Human Computer Interaction*, (forthcoming).

Nardi, B.A. (1996b). Studying context: a comparison of activity theory, situated action models, and distributed cognition. In B.A. NARDI, Ed. *Context and consciousness. Activity theory and human-computer interaction.* (pp. 69-102). Cambridge, Mass.: MIT press.

Norman, D. (1998) Affordances and Design, available at his website (http://www.jnd.org/dn.mss/affordances-and-design.html)

Reed, E. S. (1996). *Encountering the World: Toward an Ecological Psychology*. New York: Oxford University Press.

Sowa J. F. (2000). *Knowledge Representation: logical, philosophical and computational foundations.* Pacific Grove, CA: Brooks/Cole.

Susi, T & Ziemke, T (2001). Social Cognition, Artefacts, and Stigmergy: A Comparative Analysis of Theoretical Frameworks for the Understanding of Artefact-mediated Collaborative Activity. *Cognitive Systems Research*, 2(4), 273-290.