

# CSAMOA: A Common Sense Application Model of Architecture

*Jason B. Alonso*  
Tangible Media Group  
MIT Media Laboratory  
20 Ames St E15-354  
Cambridge, MA 02139 USA  
+1-617-452-5617  
jalonso@media.mit.edu

## ABSTRACT

It has become apparent that many human-computer interface applications of common sense reasoning, particularly those built on the OpenMind Common Sense corpus, make use of similar computational tools (spreading activation, for example) in addition to the corpus itself. Meanwhile, new representations, new methods of reasoning, and new applications are being introduced without a clear foundation for understanding their interrelationships. In this paper, I describe my goals and progress in the design of a model of architecture for expressing the inter-operation of common sense tools developed as parts of different efforts.

**ACM Classification:** H5.2 [Information interfaces and presentation]: User Interfaces. - Theory and Methods.

**General terms:** Standardization, Theory

**Keywords:** Common sense, application programming interface, software architecture

## INTRODUCTION

It has become apparent that many applications of common sense reasoning built on the OpenMind Common Sense (OMCS) corpus [12] make use of similar computational tools. In particular, most of these applications make use of semantic networks and perform most of their user interface magic using a small set of operations on these semantic networks. These reasoning tools, however, are usually intimately intertwined with their applications or are part of a monolithic software library, ConceptNet [9], which is a rigid framework from a software engineering perspective. In either case, new applications are bound to the development of new techniques, which hinders the research and development of both.

These tools are diverse, including such disparate components

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*IUT'07*, January 28–31, 2007, Hawaii, USA.

Copyright 2007 ACM ...\$5.00.

as raw English language corpora and spreading activation algorithms. Meanwhile, new representations, new methods of reasoning, and new applications are being introduced without a clear foundation for understanding their interrelationships. There also exists common sense knowledge representations that compete with semantic networks, and they should be accommodated. As the field of intelligent user interfaces using common sense reasoning advances from isolated experiments toward deployed applications, the need for a standard model of architecture will become ever greater. With this in mind, I set forward this draft of a Common Sense Application Model of Architecture (CSAMOA).

CSAMOA divides the software architecture of common sense applications along conceptual lines, permitting concise discourse on the roles and contributions of any given common sense reasoning application, library, or corpus as well as the development of standard Application Programming Interfaces (APIs) along conceptual boundaries. I am in the process of conducting an experiment wherein the CSAMOA model is tested by rewriting the ConceptNet database and library to fit this new form. Though the code port is not complete, I have made enough progress to draw some useful conclusions.

## REVIEW OF RELATED WORK

The projects built using OMCS do not have a consistent architecture. In this section, I review the evolution of tools leading to ConceptNet and the applications that used them, outline architectural limitations of ConceptNet, touch upon knowledge representations competing with ConceptNet, and correlate the inherent architectural challenge with a notable case from history.

## Evolving common sense tools

The OMCS family of common sense tools has been evolving and is continuing to evolve, leaving behind the applications that have built upon it. In Lieberman and Liu's Annotation and Retrieval Integration Agent (ARIA) [6], a photograph annotation and retrieval system is built upon the Common-sense Robust Inference System (CRIS), a semantic network system that served as a precursor to the ConceptNet platform [9]. CRIS was superseded by OMCSnet, which is used in GOOSE, a goal-oriented search engine assistant [7] [8]. Eventually, a suite of knowledge tools were integrated into a

single library known as ConceptNet [9]. The efforts of these projects do not benefit from the evolution of the underlying semantic network, though this disconnect is understandable given the nature of young and developing code.

Future directions of the use of the semantic network, as derived from OMCS, include OpenMind Commons [14]. Interestingly, this upcoming project, as proposed, reuses many of the ideas in ConceptNet, but the new structure focuses on using inference in real time, which contradicts the ConceptNet design choice of batch processing all assertions. Even though a semantic network will still support this new project, work must be done to keep this from being yet another divergence—this time from the ConceptNet effort.

### Limitations of ConceptNet

ConceptNet is particularly troubling from the standpoint of a software architect in that it consists of three conceptually distinct components that are inextricably intertwined:

1. a data set in the form of a semantic network;
2. a set of procedures for generating the data set from a corpus; and
3. a set of procedures for navigating and using the data set.

These three components strongly echo the well-known Model-View-Controller software design pattern. In this design pattern, the model component abstracts the data used by the software, the view component translates the model into other forms for interaction (especially human interaction), and the controller component mediates the interactions between the two insofar as controls like business logic are concerned. The segregation of roles in this design pattern improves code maintainability by allowing the effects of distinct changes to be contained.

In the case of ConceptNet, these three roles cannot be separated. Data is stored in flat text files, and those files are managed by the same scripts that store and apply regular expressions for extracting binary predicates. The scripts that permit navigation of the semantic network are not clearly separated. It is unclear, in this regard, how to describe what parts of ConceptNet are supposed to be a data set and what parts are supposed to be a software library.

### Competing knowledge representations

At the present, there is a remarkable spectrum of substantially incompatible tools in the space of common sense reasoning. These include the OpenMind Common Sense project [12], which uses natural languages as knowledge representation, and the fundamentally more ontological projects Cyc [1] and ThoughtTreasure [10]. Of the intelligent user interface applications of common sense surveyed by Lieberman et al in [7], most are based on the OpenMind family of platforms. One interesting exception is Common Sense in a Disk Jockey's Assistant (CSDJ), which made use of ThoughtTreasure instead.

### An Analogy with the Development of Informatic Networks

The early development of computer networks, both experimental (like ARPANET) and proprietary, lacked an established “interconnection architecture” to unify diverse pieces of computing equipment. The International Standards Organization (ISO) formally recognized this in 1977 and created a subcommittee known as “SC16” for “Open Systems Interconnection.” The subcommittee established, as its highest priority, “the development of a standard Model of Architecture which would constitute the framework for the development of standard protocols.” The product of this discussion was the “Reference Model of Open Systems Interconnection,” also known as the “OSI Model.” [15] This model organizes the entire gamut of communications protocols, from cabling standards to object serialization.

The history of interfaces built on the OMCS platform has clearly demonstrated that many of the components of common sense reasoning platforms are still being developed. Indeed, the OpenMind Commons knowledge elicitation platform is expected to make radical departures from the previous platforms. Furthermore, there exist resources other than OMCS for common sense, like Cyc and ThoughtTreasure, that have the potential to be useful in the same applications, but there is a great disconnect among Cyc, ThoughtTreasure, and OMCS. Even ConceptNet has competing interpretations of the OMCS corpus, like EventNet, which may be loosely described as Markov model instead of a semantic network [3].

As it stands, the development of new applications is being hindered by the limitations of lack of architecture underlying the software tools for using common sense.

### DESIGN CRITERIA

Given the correlation between the limitations of ConceptNet and the problems faced by SC16 as well as my engineering intuition, I concluded that a layered architecture similar to the OSI Model would be suitable for CSAMOA. As such, the following design goals were copied from the goals leading to the OSI Model:

1. “do not create so many layers as to make difficult the system engineering task describing and integrating these layers;
2. create a boundary at a point where the services description can be small and the number of interactions across the boundary is minimized;
3. create separate layers to handle functions which are manifestly different in the process performed or the technology involved;
4. collect similar functions into the same layer;
5. select boundaries at a point which past experience has demonstrated to be successful;
6. create a layer of easily localized functions so that the layer could be totally redesigned and its protocols changed in a

major way to take advantages of new advances in architectural, hardware, or software technology without changing the services and interfaces with the adjacent layers;

7. create a boundary where it may be useful at some point in time to have the corresponding interface standardized;
8. create a layer when there is a need for a different level of abstraction in the handling of data, e.g., morphology, syntax, semantics;
9. enable changes of functions or protocols within a layer without affecting the other layers;
10. create for each layer interfaces with its upper and lower layer only;
11. create further subgrouping and organization of functions to form sublayers within a layer in cases where distinct communication services need it;
12. create, where needed, two or more sublayers with a common, and therefore minimum, functionality to allow interface operation with adjacent layers; and
13. allow bypassing of sublayers.” [15]

My original intent was to revise the goals from the OSI Model, adapting words like “protocols” and other communications-specific terminology, but I found the original terms to be sufficiently general to apply to the development of the CSAMOA software architecture. I do, however, have one specific goal to add: *within each layer, create a clear role division between the data models and the processes applied to them.*

#### THE CSAMOA MODEL

At the time of this writing, my working draft of the stack has four layers: corpus, representation, realm, and application. In this section, I give a brief definition of each layer, followed by a discussion of the reasoning for the choice of role divisions between each layer.

##### The layers, defined

**Corpus** The first layer of the CSAMOA stack is the Corpus layer. It is the most basic component of the CSAMOA stack and is dedicated to preserving the original, human representation of common sense knowledge, irrespective of any particular application, and any notations required to preserve source information. This includes, for example raw natural language statements, elicitation frames, and source data (including user profiles) on the stored natural language statements.

**Representation** The second layer of the CSAMOA stack, the Representation layer, is dedicated to the abstraction of a corpus knowledge representation into a machine-interpretable form, like a semantic network or Markov matrix. As this abstraction is a complicated process, the Representation layer is further subdivided into the Parsing/Encoding, Reasoning, and Presentation sublayers.

**Parsing/Encoding** The Parsing/Encoding sublayer is responsible for taking the knowledge from the Corpus layer and converting its form to comply with the rest of the Representation. As there may be cases where knowledge must be converted back into a Corpus-compliant form or into natural language, this layer is responsible for performing this encoding operation as well.

**Reasoning** The Reasoning sublayer is responsible for deriving new pieces of knowledge, particularly with abductive reasoning, from the existing body of knowledge.

**Presentation** The Presentation sublayer defines the theoretical form of the Representation layer as seen by the subsequent layer, providing the requisite navigation routines and data structures for the representation of knowledge in the entire Representation layer.

**Realm** The Realm layer is responsible for navigating the Representation layer to isolate realm-specific knowledge as well as computational operations, like spreading activation, on the representation, usually without particular regard for the semantics underlying the representation.

**Application** At the top of the CSAMOA stack is the application itself, which is responsible for all user interactions and for the ultimate processing of the knowledge made accessible by the rest of the hierarchy. This layer defines the experience of the user when interacting with an intelligent user interface, insofar as the processing of natural language and common sense can be left to the lower layers.

##### Motivation and additional notes

**Corpus** It is important that much emphasis is placed on preserving accountability and human-readability for knowledge in the Corpus layer in the form of source data. This emphasis is in place to allow for ready debugging regardless of the knowledge representation (KR) in place—this, in turn, is an attempt to address the fundamental limitations of any KR as described in [2].

In the OpenMind family of projects, this role is filled by the OpenMind Common Sense corpus itself, which immediately prompts the assertion of design goal 5. In systems that have knowledge representations other than natural language, like the maps and other representations from Eric Mueller’s ThoughtTreasure [10], this layer is inclusive of these representations. One can even consider a body of CycL assertions and appropriate navigational routines to comprise the Corpus.

**Representation** The defining attribute of this layer is the form it gives to the knowledge it contains. Formally, this means that this layer defines both data structures and APIs for the computational manipulation and/or navigation of common sense knowledge. By virtue of design goal 12, I subdivide this layer into sublayers to allow a particular representation layer to be built upon a variety of corpora.

In the case of the OpenMind family of projects, the Representation layer would have been CRIS, OMCSnet, or ConceptNet [9]. The sublayers have distinct functions that are easily motivated, and I shall show how ConceptNet specified

the most developed set of interfaces out of these three examples.

ConceptNet directly included MontyLingua [9], a natural-language processing engine for English, filling the role of the Parsing/Encoding sublayer. As the parsing of the Corpus layer's KR into the Representation layer's format is a process tightly coupled to both layers, I found it appropriate to put this function into a sublayer as specified by design goal 12, rather than presume the process to be an innate feature of a Representation layer.

The Reasoning sublayer is positioned second in order to abstract away the reasoning process from an application. The intention is that the Representation layer should appear to encompass all knowledge, whether stored in the Corpus, provided by the application, or derived through reasoning. It should be noted that the Reasoning sublayer can produce knowledge that is not preserved by the Corpus layer. The knowledge used by the Reasoning sublayer may come from the Corpus, the application by way of the Presentation sublayer, or from both.

It should be noted that CRIS, OMCSnet, and ConceptNet are all semantic networks, which would be a valid specification of the data structures in the Presentation sublayer. Navigational interfaces may include methods for dumping the relationships in a given set of nodes in the semantic network or for filtering for nodes and/or relationship types germane to a particular application domain (realm filtering).

Upon close examination of CRIS, OMCSnet, and ConceptNet, it can be seen that these projects were distinguished to some extent by the number of relationship types between network nodes and the degree to which reasoning was used to generate succinct representations. Of these examples, ConceptNet was the most advanced for introducing various forms of abductive classification in what would be the Reasoning sublayer.

As all of the functions of these sublayers are intertwined, I found that design goal 2 forbids me from granting these sublayers the status of "layer," as the interactions between these layers would be diverse and numerous. As it would be difficult to make these layers highly interchangeable, I am further discouraged by design goal 6.

*Realm* The Realm layer is the most vaguely defined component of the CSAMOA stack. Like the Presentation layer of the OSI model [15], design goals 3 and 4 demand that a variety of general-purpose routines be collected into a layer just below the Application layer.

In particular, operations such as optimal path finding, spreading activation and weighting of wherein certain relationship types in a semantic network to give them critical nuances for other operations (especially spreading activation), belong in this layer to benefit multiple applications. This is probably best understood with the simple example of spreading activation. In ARIA as well as Shen, Lieberman, and Lam's scenario-oriented fashion recommendation system [11], spreading activation is applied to a semantic network. The parameters used to guide spreading activation dif-

fer in these two applications, especially in the weights assigned to different link types and the nodes chosen for initial activation.

It should also be noted that some operations, like the weighting of relationship types, are conceptual expansions of navigational operations that belong in the Representation layer. In the case of weighting relationship types, the process is an improvement of realm filtering.

*Application* It is beyond the purpose of CSAMOA and the scope of this paper to examine the modalities of human-computer interaction in systems that may be built along the lines of CSAMOA. These modalities are grouped with the all of the other remaining functions required of a common sense application into the Application layer.

## THE FIRST EXPERIMENT

ConceptNet belongs to a family of substantially incompatible OpenMind Common Sense (OMCS) -derivative representations colloquially known as the "X-Nets". Other representations include EventNet and LifeNet. It has been advanced by Smith in [13] that a desirable future direction for the X-Nets is interoperability, which indicates to me that the X-Nets should be examined closely to see if the CSAMOA model of architecture can bring a potential mechanism for interoperability to light. I also became aware of an effort to revise ConceptNet to add new features, like polarity of assertions, which was identified as a future direction for ConceptNet in [13], and the people engaged in this project were accessible to me. ConceptNet also lacked interchangeable software components, despite the changes being developed. With all of these things considered, ConceptNet appeared to be the optimal choice for testing CSAMOA.

The first experiment on the use of the CSAMOA model was restricted to ConceptNet and the underlying OpenMind Common Sense (OMCS) corpus. This limited my ability to search for interoperability with the other X-Nets, as such a search would require the close examination of at least two X-Nets, but this limitation allowed the first experiment to test CSAMOA for less esoteric benefits. The benefits sought in this experiment were:

1. a clear separation of dependency relationships among the common sense software components; and
2. intuitively obvious assignment of any data model or procedure to an appropriate layer.

It must be noted here that, though interchangeability of software components *may* be enabled by CSAMOA, I do not expect enough alternative structures to be developed in the course of this experiment to test CSAMOA's capacity for it.

## THE EXPERIMENTAL IMPLEMENTATION OF THE MODEL

In the case where all layers of the model need to be implemented, the model naturally lends itself to a bottom-to-top construction, starting with the Corpus layer. Encouraged as such, I began my implementation there and worked toward

the Realm layer. It should be noted that, by itself, ConceptNet does not complete an application, so I do not expect to populate the Application layer with any code.

The original ConceptNet code is written in the Python [4] interpreted programming language. Given personal preferences and the fact that the ConceptNet redevelopment effort with which I was cooperating was continuing to use Python, I opted to keep the chosen language for this experiment the same. I anticipated the appropriateness of a strong database abstraction layer, wherein database tables and queries are generated automatically from naturally-written Python code, and so I selected the Django web development framework [5].

At the time of this writing, however, the conversion of the ConceptNet code to the CSAMOA model was not complete. I made it through the development of the Corpus layer and much of the Representation layer, with some conceptual work throughout the remainder. The particulars of my progress and observations follow.

### **Corpus**

In the Corpus layer, I implemented models and control code to manage the OMCS corpus. Though the code developed is essentially complete and functioning, it does not include any knowledge elicitation code as would be required to have a complete OMCS system [12].

In the original ConceptNet code, an autocorrector for fixing common typographical errors was included in the code that parsed sentences. As this operation produces data in the same form as the original data in the corpus, I determined that autocorrection code should be separated from the parsing code and assigned entirely to the Corpus layer. This code became a generalized autoreplace engine, with distinguishable rule sets, when I discovered “swaplists” in the ConceptNet code with subtle semantics-altering effects deeper in the parsing code. It should be noted that the “swaplists” were applied indiscriminately in the parsing layer, such that it would be equivalent to put the substitution rules in the Corpus layer.

The database models that I created for this layer include the supporting elements Language (for keying multilingual dictionaries and rule sets), AutoreplaceRule (for providing the aforementioned autoreplace rule sets), and Activity (for representing the knowledge elicitation method). The layer also includes the core elements Source (for storing profile data on the provider of a piece of common sense knowledge) and Sentence (the smallest unit of corpus data).

### **Representation**

The Representation layer, as anticipated, contains the bulk of the ConceptNet transformation. The models were easy to construct, with some exceptions that will be discussed.

*Parsing/Encoding* ConceptNet used a number of parsing dictionaries for storing word sets like stop words and words that invert meaning—this was easy to accommodate, but the use of these dictionaries was so far removed from the publicly-exposed models of this layer that I was encouraged to place them into their own “tools” subcomponent of the Parsing sublayer. After all of these preprocessing

steps, ConceptNet chunked and tagged its sentences with the MontyLingua natural language toolkit, which I expect to be easy to adapt and/or replace in this reimplementation.

Models that I created in this sublayer include the defining models Frame (for storing elicitation frames) and RawPredicates (for storing binary predicates extracted from a Sentence).

*Reasoning* The Reasoning sublayer is dominated by the reasoning process, and no supporting models, like dictionaries, have warranted themselves necessary yet. The algorithms in this sublayer are being redeveloped by Speer [14], but the work I have seen so far seems to indicate that the Reasoning sublayer can rely entirely on the models provided by the Parsing and Presentation sublayers with their respective tool kits.

*Remaining layers* My work in porting ConceptNet is not yet complete, though I consider it important to note that there have been no major conceptual or technical setbacks. That is, with CSAMOA in mind, the porting of ConceptNet has proven to be a straightforward engineering effort thus far, requiring only additional time.

### **FUTURE DIRECTIONS**

As the experiment described in this paper is not complete, I plan on finishing the transformation of ConceptNet into the CSAMOA model, using GOOSE [8] (the goal-oriented search engine assistant) or Empathy Buddy [7] (the common-sense-driven affect-sensing system) for the Application layer. CSAMOA should be further evaluated with the reimplementation of LifeNet or EventNet as alternative Representation layers, probably with Markov matrices instead of semantic networks.

While CSAMOA is not an API that can naturally translate OMCS sentences to and from another KR, like CycL, CSAMOA as a model of architecture can facilitate the development of applications that use multiple underlying KR in tandem—this is an interesting potential future direction.

I would also like to seriously consider changing the name of the Representation layer to be the “Interpretation” layer, so as to reduce the possibility of confusion with the meaning of “representation” in the term KR as used in [2].

### **CONCLUSIONS**

CSAMOA, though not fully tested at this time, is demonstrating great potential as a software architecture for common sense applications. It was designed with the approach to the OSI Model, a historically significant example of good engineering, as a basis of design, and an ongoing reimplementation of ConceptNet appears to be benefiting from this decision.

It remains difficult to measure the successfulness of CSAMOA as a taxonomy for common sense software components, though the naming of various components of ConceptNet, LifeNet, and EventNet has felt natural in the development of this project.

## ACKNOWLEDGMENTS

I would like to thank Rob Speer and Catherine Havasi for their patience and understanding as I scrutinized everything, and continue to do so, while they worked on reviving the ConceptNet code. I would also like to thank Henry Lieberman, Junia Anacleto, and Dustin Smith for their insights on the structure and future of ConceptNet.

## REFERENCES

1. Inc. Cycorp. How does cyc reason? Cyc web site, 2006. [http://www.cyc.com/technology/technology/whatiscyc\\_dir/howdoescycreason](http://www.cyc.com/technology/technology/whatiscyc_dir/howdoescycreason).
2. Randall Davis, Howard Shrobe, and Peter Szolovits. What is a knowledge representation? *AI Magazine*, 14(1):17–33, 1993.
3. Jose Espinosa and Henry Lieberman. Eventnet: Inferring temporal relations between commonsense events. In *Proceedings, Fourth Mexican International Conference on Artificial Intelligence*. Springer, 2005.
4. Python Software Foundation. Python programming language. Python web site, 2006. <http://www.python.org>.
5. Lawrence Journal-World. Django, the web framework for perfectionists with deadlines. Django web site, 2006. <http://www.djangoproject.com>.
6. Henry Lieberman and Hugo Liu. Adaptive linking between text and photos using common sense reasoning. In *Conference on Adaptive Hypermedia and Adaptive Web Systems*, LNCS 2347. Springer, 2002.
7. Henry Lieberman, Hugo Liu, Push Singh, and Barbara Barry. Beating common sense into interactive applications. *AI Magazine*, 25(4):63–76, 2004.
8. Hugo Liu, Henry Lieberman, and Ted Selker. Goose: a goal-oriented search engine with commonsense. In *Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, LNCS 2347, pages 253–263. Springer, 2002.
9. Hugo Liu and Push Singh. Conceptnet: a practical commonsense reasoning toolkit. *BT Technology Journal*, 22(4), 2004. Kluwer Academic Publishers.
10. Erik T. Mueller. Thoughttreasure: a natural language/commonsense platform. Signiform web site, 1998. <http://www.signiform.com/tt/htm/overview.htm>.
11. Edward Shen, Henry Lieberman, and Francis Lam. What am i gonna wear: Scenario-oriented recommendation. In *IUI '07: Proceedings of the 12th International Conference on Intelligent User Interfaces*. ACM Press, 2007. Submitted.
12. Push Singh. The public acquisition of commonsense knowledge. In *Proceedings of AAAI Spring Symposium: Acquiring (and Using) Linguistic (and World) Knowledge for Information Access*. AAAI, 2002.
13. Dustin Smith. The future of commonsense reasoning. Presentation, March 2006.
14. Rob Speer. Openmind commons: A new framework for acquiring common sense knowledge. M.Eng. thesis proposal, 2006. <http://torg.media.mit.edu/rob/index.php/Research>.
15. Hubert Zimmermann. Osi reference model—the iso model of architecture for open systems interconnection. In *IEEE Transactions on Communications*, volume 28, issue 4, pages 425–432, April 1980.