

1. Table of Contents and Figures

2. Identification and Significance of the Innovation3

 2.1. For NLP Workers – an Open-Source Context Management Utility

 2.2. For Modeling – Topics Networked by Executable Case Frames

 2.3. Our Agency Web Site – On-line Charting as a Service..... Fig 1

 2.4. Customization by Application-specific Lexicons and Interface Agents

3. Phase I Technical Objectives 9

4. Phase I Work Plans12

 4.1. Open Source Components will be IntegratedFig 2

 4.2. Add Discourse Models: a list of Recently Accessed Topics

 4.3. Formally Define the WORDS Language Using a Published Ontology

 4.4. Add Operator I/O: a Desktop DIALOG User InterfaceFig 3

 4.5. The Chart Models and Connects Stages of Paragraph Analysis.....Fig 4

 4.6. Define Example MEANINGS in the CFT and Extension Lexicons.....Fig 5

 4.7. Our Phase I Research Report and Proposal for Phase II

5. Related Work 17

 5.1. Our Semi-Deterministic English Parsing System..... Fig 6

 5.2. Continuing R&D on Topic Maps and CONTEXT

6. Relationship to Future R&D..... 19

7. Commercial Applications Potential..... 21

8. Key Personnel & Bibliography 22

9. Company Information and Facilities.....23

10. Consultants and Sub-Contracts23

11. Related Proposals to other Federal Agencies24

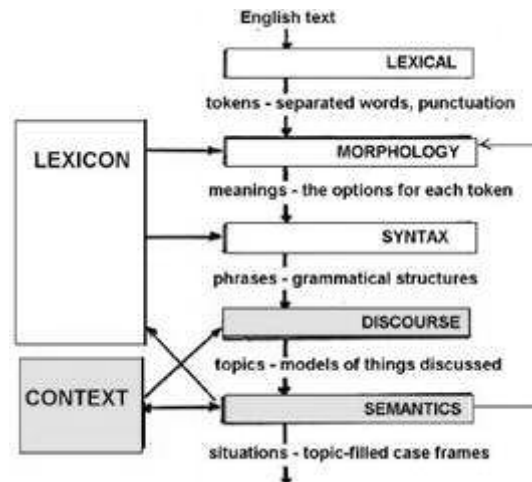
12. Bibliography24

2. Identification and Significance of the Innovation

English paragraphs are the most common form of recorded information, yet today’s software cannot gain access to their meaning. Lexikos has devised a way to change that with a desktop tool that can semantically interpret the clauses and phrases of grammatical English text.

If loaded with suitable lexical data modeling word semantics, this tool can semi-automatically reformat a paragraph into portable XML metadata that accurately models its contextual topics and whatever was said or asked about them. Such metadata models dramatically improve the potential for human-automation interaction in information retrieval, message routing, situational awareness, model-based problem-solving or tutoring, embedded training, and similar hard tasks.

In classical NLP (Natural Language Processing) technology, our interpreter would be a utility that builds and uses *SEMANTIC* and *DISCOURSE* models to help some parser analyze text. Indeed, when we first started work on this software many years ago, we envisioned it in just this way. The diagram below shows it as the bottom three modules in a typical text-analysis chain:



As our designs matured, these modules have increasingly seemed useful as a new product in its own right – one meant for high-level semantic modeling – with the parser only a support process, or even optional. As this line of thinking grew, a surprising number of applications for such a new tool have emerged, touching several fields that seek to process symbolic knowledge.

As the back end to a parser, our design uses textbook NLP concepts on “logical forms” [17], which are rare but not novel. As an independent tool, however, new use cases led to changes. *CONTEXT*, for instance, evolved from an inference engine into a simpler engine for a Topic Map, the ISO-standard metadata paradigm [21]. Its XML interchange format can then output paragraph models, and also initialize *CONTEXT* with the semantic lexicons for any domain.

Accessed through a desktop UI, our web-based tool is significant to the goals of this project. Used like a simple spelling corrector as each new paragraph is authored, it can not only repair misspellings but simultaneously generate linguistically accurate metadata – in what we call a *chart* - on all topics of scanned input text, and on all assertions made about those topics.

The output chart is standard XML - easy to pass about, store, encrypt, and (for software at least) read. But the specific metadata models we put into it make the input text *understandable* by application software. It can query the chart, analyze and transform it further, or route it by email to someone who has asked to learn about paragraphs on the sort of topics it models.

Our standard chart is not precisely the metadata you envisioned, for by design our tool is general. But if you integrate it properly into your operations, and help users build up the custom lexicons upon which its accuracy depends, then each chart (see 4.5) will indeed be easily converted by a central automatic process into just the metadata you require. Further, it will support *additional* software for distributing the related intelligence data rapidly to everyone who needs it. Our tool – more fully explained below - thus trims the core problem down to (almost) a routine IT task.

2.1. For NLP Workers – an Open-Source Context Management Utility

NLP research has produced innumerable parser designs and examples. Simple ones are assigned as homework when an Introduction to AI course hits the NLP section. Commercial versions are tougher, as they face complexities such as conjunctions, elision, idioms, movement effects, etc. Worse, they must cope with bad data - misspellings, slang, dialects, unknown words or names – and find partial parses if their workarounds fail. All this takes many man-years of costly R&D.

Lexikos delivered such a full-featured parsing system in 1990, under SBIR phase-II funding [1] - a broad coverage, Common Lisp implementation of an enhanced Marcus Parser [29]. On syntax, it worked great. But for funding reasons, ours had to omit the separate machinery needed to interpret textual meaning – a “*Context*” module able to do a semantic and discourse analysis.

Despite a nice deterministic design in its *grammar*, this forced our parser to use special tricks (trial and error, operator help) to handle certain related linguistic effects – like competing word senses and attachments for prepositional phrases. Such hacks clobbered its run speed, but even worse, they severely limited its applications. We *needed* a semantics-and-discourse package.

The design we sought – which phase I of this research plan will finalize – is a semi-independent module which can interpret any stream of syntactically localized (parsed or marked up) English phrases and clauses. It uses a **literal** reading of what they objectively depict in the context of the discourse – and reports known semantic constraint(s) violated by such a reading. Specifically, this lets our interpreter build up persistent clause-by-clause semantic models which...

- Model what things were discussed – the topics overtly mentioned (often repeatedly)
- Formally express (with limited validation) the assertions made about each topic
- Show the input text unit(s) and speaker(s) responsible for each semantic feature

Such models are significant because they are rare. They can usually be encoded only AFTER a huge base of other complex and expensive R&D has been done, and few organizations persist long enough to get into them. But they do deepen paragraph analysis, by letting a parser:

- Locate the referents for anaphora – pronouns or definite noun phrases
- Properly attach prepositional phrases to whatever they modified
- Pick the intended senses for words within (and after) modeled text units

Our planned Java software can work standalone with specially parenthesized text inputs. Or for real voice or text inputs, it can be an open-source download extending multiple syntax analyzers – not just our own. It can let them and their applications address English paragraphs using not merely their syntax, but topic and semantic models – a major improvement.

Such changes foreshadow others involving speaker intentions; long-term learning rules of conversational courtesy, metaphor; etc. These effects demand more common-sense knowledge than Lexikos can add alone, but with help from our customers and resources like OpenCyc [18], they will emerge. Meanwhile, being able to address meaning at any level will let NLP code do a better job on sentence structure, and that alone is ample to justify our interpreter’s R&D.

2.2. For Modeling – Topics Networked by Executable Case Frames

To more clearly see the *Knowledge Engineering* significance of *Context*, it is best to first step back from details, and look at the “big picture” flow of the information passing into it:

1. An author or speaker notices X - something real, to be modeled linguistically
2. He or she then creates a linguistic representation of X using a stream of words
3. A parser uses a lexicon to convert that stream of words into lexical data models
4. Its rules, in a grammar, filter and arrange those lexical models into parse trees
5. Our interpreter then maps such parse trees into “semantic” models in *Context*

If everything works, the data saved at step 5 will decently model *any X* noticed at step 1. Logical forms and discourse models and other NLP conventions may help formalize the notation, but at its core, *Context* has to model *things*, not just the language used. The true goal of this research (given sufficient lexicons) is that our design for *Context* and its Interpreter must do this in the general case – at least superficially - for any X a human might describe in English!

Among the best-known strategies for handling simple semantic models and constraints in NLP are *Case Frames* – an approach popularized by Charles Fillmore [3]. In this paradigm, patterns of expected complements for verbs and other words get used to predict and limit the structures which a parser actually assigns. Fillmore’s Berkeley FrameNet project is now making public this vital data in bulk. Large complementary projects are also ongoing, notably PropBank [32].

In prior work, Lexikos has assembled its own base of lexical data for NLP, covering all common English verbs, irregulars, and complex modifiers. We took the time to hand enter detailed usage data on each word, including specific patterns of complements taken by the complex word types, and semantic markers based on *Roget’s Thesaurus* [16]. By using them, our suitably enhanced code could retrieve any relevant case frame templates via straightforward lookup heuristics.

Assembled as we plan it, all such data is pure gold to a parser, as it effectively says what text structures to seek next in the input stream and how to interpret them. This impacts knowledge representation too, not only by showing *how* one might do it, but also *what things most need representing*. There turns out to be one vital set: all the situations named by English verbs and prepositions. These are exactly what case frames depict – and all English sentences too!

Case frames alone are not quite enough; discourse models are also needed. To see why, note that if any role in a situation model were filled by a pronoun, its lexical data would add nothing new. So to fully exploit case frame semantic models, we must use topics – modeling the *referents* of phrases, pronouns, and other anaphora – not just mere words. Our discourse model must track recent topics, so a parser can find these models and refocus attention [22] on them as needed.

Our *Context* design integrates both kinds of effect – case roles and discourse topic models – into one evolving data store. Able to capture and record semantics for any dialogue or document, this dynamic *data store* is what we need to model text’s meaning. It is what gets queried and incrementally mutated by our interpreter, via formal, high-level commands to its API.

2.3. Our Agency Web Site – On-line Charting as a Service

Lexikos faces business and deployment constraints as well as scientific ones. Commercialization of the software we plan means it must be able to run on-line, so that we can offer demos and training; let users try our tools before buying; solicit new partners, investors, and clients; etc. This diagram shows our plan – a web site we'll construct and alpha test by the end of **phase-II** development:

The AGENCY – This web site is the hub of a client-server system. A client typically uploads an English paragraph. A *chart* file is then returned, holding a linguistic analysis from the server-side elements, which work together to build up this return in several layers:

The PARSER – This structures input text (see Fig 6) by diagramming the syntax of each sentence and phrase in a given paragraph. This step exploits the working base of Lexikos software we have *already developed*, including our lexicon, grammar, scanner, etc. But before we can use them on-line, they must be updated to integrate properly with

The INTERPRETER – Under a web API, users configure this with WORDS models specific to a named corpus. These extend a new Lexikos database - *The Case Frame Thesaurus* – which we will test, refine and extend in this project. Via other commands, users or our parser can then create, link, and query linguistic associations of topics (Java objects) that represent and model the referents of parsed English syntactic structures.

CONTEXT – Independently web-accessible, this *data store* associates semantic models for all topics of the named corpus, attached to each client's web session. It evolves under inputs from the other modules above. To guide programmatic access, it internally builds a semantic *chart* of each new paragraph modeling its topics and assertions (see Fig 4).

Interface Agent – a downloaded Java or .Net client, able to use the above site as a remote utility to parse text inputs for its local user, then *react to* the returned *chart* in semi-intelligent ways. We foresee many such IAs may someday exist, each aiding a specific application, and acting as a dynamic interface between it, a local source of text, and our charting engine.

Figure 1 - The Agency Web Site – Server Side Components

Each IA generally supports a “training” mode that lets its human operator interact with on-line Agency code as it works - to repair misspelled words, define new vocabulary, confirm parsing heuristics, etc.. Use these options well, and the Agency output can become extremely accurate!

Training takes operator time, but it can be a great *enabler* for many projects. With modest effort and cost, a sponsor or client can tap into our extensive web-based thesaurus of *case frames* for verbs, prepositions, and modifiers, and use them on line in a session-specific Context. Once Agency learns the application-specific nouns and names, it can usually run fast and unaided.

Eventually, we will license clones of key portions of *Agency* as a portable web app. Individual customers can then run it privately to extend their own parser, lexicons, and/or applications

2.4. Customization by Application-specific Lexicons and Interface Agents

A downloaded Interface Agent is specifically meant to help a trained but non-technical user to input and parse English paragraphs, then display application-specific results. Different interfaces will exist, easy to customize in open source Java. For a typical example handling dialog Q&A...

- A paragraph of text (possibly **very** brief if a query) is collected from the user, who enters it into a U/I form (via typing, cut-&-paste, reading it from the web or a local file, etc.).
- The IA posts it to our Agency web app, which returns a *chart* (analysis) of the input text
- IA code unpacks this *chart* and passes its data selectively on to a “target” application available locally, which returns a textual response stream (the answer) back to the IA
- The IA displays it with parts of the chart, then loops to get another input from the user

Notice that our IA is actually not very smart. All it knows how to do is get English text, obtain a *chart* for it, then pass on selected semantic and discourse data from it to the (arbitrary) API of the “target” software. Most real intelligence actually resides in the things to which it interfaces.

A key interface is to a human operator, who guides the IA toward specific inputs, defines output folders and corpus-specific extension lexicons, and selects the kinds of operator-interaction to be allowed. The semi-standard *training mode* asks of its operator only well-selected input text and clerical-level skills. Under it, the IA works like a spelling corrector, asking its user questions on confusing words shown in context, learning new names and specialty vocabulary, and generally moving through a text as swiftly as it and the operator can jointly manage. Misspellings (a very common effect) degrade overall throughput. If one turns off all interactivity, run speed will be quite fast, but chart quality will decline. Users must manage this tradeoff as they see fit.

Extension lexicons built earlier in training mode play a big role, by letting the *chart* correctly cite meanings for **all** words and names in a corpus. Without them, Agency would know only general word senses and miss much of the specialty vocabulary used in real-world texts. Most specialty words are nouns or names with fairly simple linguistic features, easy to define with a keystroke or two [1]. Knowing the corpus name lets an IA scope such incremental enhancements.

For more complex additions of situational models or complex nouns, a process will be supported to merge in text files. And in phase II, we will explore a feedback loop from *chart* to specialty lexicons, such that features of words can be inferred from examples of actual usage.

In phase I, we focus instead on *standalone mode*, in which a subject matter expert adds semantic models hand-entered or read from a file directly in WORDS (no parsing). The exact rules used here need to be refined on the basis of experience. Example lexicon files we write for the CFT itself start this process. They need to be collected for ease of review and sharing, both on the Agency web site itself, and also within an open-source repository where other knowledgeable WORDS users can contribute models related to specific subjects.

Lexikos asks for your help in getting this work underway, by sponsoring our work on Phase I.

3. Phase I Technical Objectives

Our main Phase I technical goal is to demonstrate feasibility for the integrated web site depicted in figure 1, and our interpreter for **WORDS**, by prototyping and analyzing these elements:

- A. **DIALOG** – a Java application able to serve as the UI module for **MODELER**
- B. **MODELER** – a web app to build, find and modify topic models in a **CONTEXT**
- C. **CONTEXT** – a knowledge base modeling topics of English phrases and clauses

We will also migrate under a separate Java web app at least the first of these Common Lisp modules, and analyze the best options for integrating them all with modules A through C:

- D. **SCANNER** – a lexical analyzer (see fig 5) able to turn English into “lexeme” models
- E. **PARSER** – extensions (see fig 6) able to map such lexeme models into parse trees
- F. **TRANSCRIBER** – new code able to map such parse trees into **WORDS** commands

Overall, our final report will cite our prototypes as props, discuss problems we met and overcame, and answer the key questions enumerated below as sub-headings.

3.1. Can Scripted Topic Maps Support a Viable CONTEXT Model?

Lexikos spent many months (see 5.2) selecting specific tools to support a good working model of **CONTEXT**. Our storage choice is Topic Maps [20], an emerging metadata standard from ISO, with a good interchange file format [XTM, 21], multiple vendors; a dedicated open-source user community; smart consultants on email lists, and several conferences a year. Topic Maps also can merge, so any future charts and lexicons we record in **CONTEXT** can be easily combined.

Merging feels like a good start. Topic Maps have several other features we like, but the truly central one is a power to model Topics not just with String properties, but by associating them in patterns that resemble case frames or the abstract graph of a more complex scene – like some person imaging himself in a restaurant. Typical data stores might manage such models in theory, but not readily. But people can model them easily using English, and so can Topic Maps.

We have published specs [5] on how users can control these powers in **MODELER** by using **WORDS** scripts and procedural knowledge-representation techniques - data inheritance, default values, change constraints, association templates, and similar elements of a modeling framework more specialized to language than standard Topic Maps, yet still compliant with normal Topic Map paradigms. Goal one for us to show that these extensions work, and that they really do make a memory system which can readily hold, update, and associate Topic models in flexible ways that resemble what linguistic theorists might expect of a discourse model, case frames, etc.

That is a sizeable job, and it must work first at the storage level, independently of any high level notation **WORDS** defines to let users build up such structures symbolically. Answering question 3.1 is thus important not only as a baseline for further work, but for props in our final report that show what data structures support **WORDS**' associative, aspect-oriented modeling style.

3.2. Will Our Case Frame Thesaurus Usefully Model Complex Situations?

MODELER will interpret scripted models of vocabulary we call TERMS and MEANINGS. As their names indicate, these data structures loosely emulate structures in a printed dictionary. By modeling a main entry and related senses, each TERM *simulates* the intuitive notion of a *word*.

MEANINGS are their sense models – aspect-oriented models of case frames with constrained roles able to be filled. Such models encode one defined type of *situation* with all of the *parts* (its case roles) that logically relate. This is far less fuzzy than one might fear. For the BUYING concept, for example, one always expects a BUYER, a SELLER, a PURCHASE and a PRICE.

Our CFT models many such concepts, but verb MEANINGS are the easiest to visualize. The Topic Map rules define *associations* that make their patterns easy to depict, and our ontology [5] lets us customize templates for them with constraints on expected roles. For example, BUYER and SELLER topics above should always be marked as *sentient* (a smart person or a group).

A key trick to modeling semantics efficiently is to define general concepts first – e.g., make BUYING (and SELLING) a kind of EXCHANGE. But concepts of commerce are among the most complex in English, so we must work up to them gradually. For simplicity, the “building block” patterns below must all come first, so our CFT can expand in sensible ways. If we show them all as we plan, and show we can combine them, it will validate our whole approach:

- Special classes of concept often hard to model as *objects*, including
 - Substances – in English they are mass nouns (not countable)
 - Abstractions – concepts involving time, sequence, order, quantity
 - Collectives – groups of things or measured amounts of *stuff*
 - States of Being – may involve attributes or relationships
 - Changes – a whole paradigm exists for Actions, Events, Processes
 - Situations – predictable collections of related, dissimilar things
 - Aspects – the properties or parts of anything that make it unique
- Vitally important concepts involving linguistic Relationships
 - The whole-part relation
 - The class-subclass relation
 - The class-instance relation
 - The cause-effect pattern
- A temporal modeling framework for dealing with time and tense
- A spatial relation framework for dealing with 1, 2, or 3 dimensions
- A descriptive framework that handles symbols as references to things
 - The name-referent pattern
 - The word-meaning pattern
 - The model-modeled pattern

I assign no TERMS (yet) to the MEANINGS above. English has a curious way of letting *many* TERMS with different nuances and parts of speech all share the same basic MEANING. A look in Roget's Thesaurus [16] will show this. (We'll come back to this specific idea later on.)

3.3. Is Our Desktop WORDS Interpreter Intuitive to Agency Operators?

Exploring this is really our main goal for Phase I research. To do so using our CFT prototypes, we want to test, critique, and refine all extra machinery we add to a normal Topic Map, including its ontology, CONTEXT, and the WORDS scripting language that MODELER interprets.

The object of the game – and it *really is* similar to a word-game – is to identify rules that turn simple, formal WORDS commands into an assembly of typed, validated topics interlinked within CONTEXT. The data structures should model semantics of one English clause, and associate topic models with each other and its phrases such that the whole model “makes sense”.

The true test of our topic structures however, is not merely esthetic, but something more precise: if they are later queried, can they return data which shows an unambiguous model of the surface semantics of the original clause? If so, then the meaning of that clause was logically embedded somehow in the data structures, so that test passes. If not, we have more work to do.

MODELER is the framework in which all such activity occurs and DIALOG is its UI module. Any UI is very important, especially when design goals like “intuitive” get raised. In addition to the WORDS language itself (which must reuse English grammar), the UI in which it is entered must send very clear signals on what is going on, and what is expected next of the user.

DIALOG collects from a user some symbolic queries, assertions, and/or commands that one can express in WORDS, then passes them along to MODELER using the same API that a parser would employ. This last requirement is crucial. It lets DIALOG and some rules on parentheses *take the place of a parser* during our Phase I work, so that we can focus mostly on modeling issues, unit tests and debugging, not on cross-language software integration issues.

DIALOG must keep a WORDS user informed about the current state and recent changes in the discourse and semantic models within CONTEXT. As part of its operation, it also handles such administrative tasks as user log-in, plus the saving or restoring of all user-specific data. In all these areas, it acts as a first working prototype of an Agency IA and sets their design pattern

3.4. What Plan Best Arranges Phase-II Integration with TRANSCRIBER?

SCANNER and TRANSCRIBER – systems which WORDS users will effectively hand-simulate – today exist as Common Lisp code. With no user interface that would work on an Agency web site, they also have no interface (yet) to the Java API which MODELER will expose.

These interface problems are related, and both need to be solved at once, not just for the specifics of our own parsing logic, but as much as possible in the general case, so that our MODELER code can be easily used by the parsers of future customers and sales partners.

Such software engineering questions are remote from linguistics, but vital to our success. We will resolve them as a support task, planned for a specialist member of the Phase I R&D team.

4. Phase I Work Plans

Developing even a game-like an interpreter for English sounds ambitious, but our design carefully leverages a ton of excellent past R&D by many talented people, inside Lexikos and out. We will integrate it as a baseline for our prototype interpreter, which runs in this architecture:

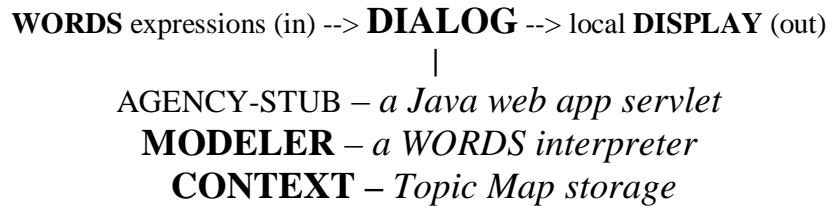


Figure 2 - The Interfaces in MODELER

4.1. Open Source Components will be Integrated

We will download the latest Java code for two loosely licensed open-source Java packages. Huge practical contributions to our goals are made by this cost-free support software:

- **JSCHEME** [6] is a lovely piece of freeware originally coded by Peter Norvig, who literally wrote the book [12] on modern AI. It is an interpreter for Scheme, written in Java, which can call Java methods and be called by them
- **TM4J** [14] is a free, open source Topic Map engine, able to handle their Topic Map storage, mutation, import, export, merging, and queries. It can store data as transient Java object, or for persistence map them into a number of standard data bases.

4.2. Add Discourse Models: a List of Recently Accessed Topics

To support our UI and help organize CONTEXT internally, we must add machinery that WORDS can use to maintain a discourse model. Initially, this can be quite simple – a list of referenced Topics, kept ranked by their most recent access within the input stream.

Using an “MRU list” like this is an old NLP trick, but effective for handling anaphora – a pronoun like “she”, or a definite noun phrase like “the woman”. In either case, to resolve the reference - to *find* the indicated Topic in CONTEXT - the discourse logic can simply search its MRU list for the first Topic to exhibit the implied constraint of “femaleness”. These examples help clarify how WORDS can emulate English by using relatively simple rules:

- (a woman with (height tall))
- (the woman with (height tall))

The first WORDS expression above is an indefinite noun phrase, which causes MODELER to **build** a Topic for a tall woman within CONTEXT. The second reference is definite. It causes MODELER to **find** the first tall woman Topic on its current MRU list.

4.3. Formally Define the WORDS Language Using a Published Ontology

[Budget: 4.1-3 will take about 8 weeks by PI, aided by Tech writer, to meet Goal 3.1]

In a classic AI technique, we “embed” WORDS in Scheme to reuse that language’s existing syntax, macros, primitive functions, documentation, control-flow, and support machinery - a big leap forward! WORDS then must add only functions, macros, symbols, and templates:

- **TOPICS** – Some WORDS expressions make MODELER find or build Topics, using our discourse module and TM4J. Their equivalents in English are special symbols (names and pronouns) or else noun phrases – captured with expressions like those in 4.2. In all such cases, the WORDS expression returns a Topic stored as a Java object in the JVM.
- **PHRASES** – Any list of WORDS expressions headed by an English preposition or verb is used by WORDS as a spec for an *association* [26] of following Topics. Details depend on a template pre-stored for the leading predicate. By supplying templates that resemble case frames, our CFT can control in fine detail the specific associations to be used.
- **CLAUSES** – a sentence or relative clause in WORDS just pushes a Topic expression for a subject into a verb phrase, whose association then includes that subject Topic, which is returned. Similar logic used for prepositional phrases *attaches* them to a subject Topic.

The devil is in the details, but these rules summarize our core design for WORDS, and illustrate how association templates act as digital definitions for verbs and prepositions. Our templates for them (see 3.2) represent case frames for situations – a *semantic* model, not a syntactic one! And they can hold scripts that check and/or modify any Topics assigned. By using such tricks, MODELER lets semantic models for English text be built in bulk from WORDS expressions.

4.4. Add Operator I/O a Desktop DIALOG User Interface

DIALOG implies a conversational mode of UI - mixed initiative dialogs in which a user leads but code may interrupt to ask questions. The right panel below shows recent Topics in CONTEXT, and guides the operator on using anaphora: [Budget: a UI consultant helps us meet Goal 3.3]

<p>Response(s) of Client Agent to Previous Operator Input</p>	<p>Links to Models of Topics in Focus</p> <p>·</p> <p>·</p>
<p>TEXT FORM</p> <p>(The Operator Adds Next Input Here and Submits)</p>	

Figure 3 – The “Dialog” input form displayed

4.5 The Chart Models and Connects Stages of Paragraph Analysis

The main output of the *Agency* web site is an XTM file we call the *chart*, whose layers get added sequentially as any English input paragraph gets processed. Like a digital version of the Rosetta Stone, each *chart* holds several languages with related content:

#0: Header data from the GENERATOR	The time and location that layer 1 was processed, and its human source(s). Extra application-specific data may be added. Other layers below accumulate as different linguistic modules report their analyses..
#1: English text from the operator	The original input paragraph acts as auto-documentation for humans, and summarizes all the remaining layers, which are written in languages meant for machines.
#2: Lexeme map, from SCANNER	This would hold data on individual words and names in running text, showing roots, inflections and lexical data. A partial example for one paragraph appears below in Figure 5
#3: Topics from parse tree phrases	A parser would find these syntactic units by parsing the stream of input lexemes in level 2. A noun phrase gets recorded as a WORDS expression that returned the ID of some Topic, either found or built
#4: Sentences and Clauses (cites #3)	Sentences can be “diagrammed” in a WORDS expression that lists Topic IDs from level #3, associated grammatically by prepositions, verbs and functional words
#5: Speech Acts, from the rules of DIALOG	MODELER can offer advice that would let an IA <i>respond</i> to each item in level #4 by using default response rules, but each target application can easily ignore or override these suggestions.
#6: Optional reply from an IA client	The application code reaction to levels #0-5 (which are effectively one <i>Agency</i> input command) can be optionally logged here. A visible reply can also be added, which the IA will display to the operator

Figure 4 – The Chart for each Paragraph after Local Processing

This abstract *chart* models sequential stages for Phase II analysis of a paragraph. In phase I, we will investigate building it under the XTM 1.0 DTD [21] to enable communication of parsing data and results to our IA client [Budget: This sub-contract – see 9.2 – helps meets Goal 3.4]

Figure 4 makes it easier to see how the various processes interact and build on one another during the analysis of a paragraph. On the next page, Figure 5 shows output from our current SCANNER, after lexical ambiguity has been artificially lowered by an operator. It could be passed at level #2 to any parser, whose WORDS commands to MODELER would then lead to creation of levels #3 and #4. The last levels would involve client-side code.

Can SCANNER use the MRU-list and partial associations in CONTEXT to heuristically cull lexical ambiguity from inputs **without help from an operator**? If so that would be a practical advance in NLP technology (and big news for information retrieval and voice recognition). Our phase II system and charts will support test results for heuristics meant to enable this.

"Many Pentagon experts believe the Aegis is being called upon to do something in the Persian Gulf for which it was never really designed. It is designed to operate in open waters and simultaneously track hundreds of targets, differentiating friend from foe. It is not designed to work in a very small, mixed civil and military aviation environment and to tell commercial flights from military flights.

"

(MANY (QUANT PL (MEANS *NUMEROUSNESS)))
(PENTAGON (NAME ORGANIZATION))
(EXPERTS (NOUN PL HUMAN (ROOT EXPERT)
(MEANS *PARTICULARITY *SPECIALTY *KNOWLEDGE *SKILL
*ADVICE)))
(BELIEVE (VERB BARE INTRANS TRANS THAT-SENT-C SENT-C NP-INF-C
Q-C NP-NP-C NP-ADJ-C NP-PASTP-C
(MEANS *THEORY *BELIEF *SANCTITY *PIETY)))
(THE (DET SG PL MASS DEF))
(AEGIS (NOUN SG))
(IS (AUX 3P-SG (ROOT BE)))
(BEING (AUX PRESP (ROOT BE)))
(CALLED (VERB PAST PASTP PP-INF-C (PREPN UPON) (ROOT CALL_UPON)
(MEANS *COMMAND *DEMAND *ETHICS *DUTY *IMPOSITION)))
(UPON (PREP IDIOMATIC))
(TO (COMP))
(DO (VERB BARE TRANS INTRANS IO-NP-C NP-AS-NP-C
(MEANS *STATE *AGREEMENT *CAUSE *PRODUCTION *PRODUCT
*PRESENCE *SUFFICIENCY *USE *ACTION
*ACCOMPLISHMENT *BEHAVIOR)))
(SOMETHING (PRO SG NEUT INDEF PRE-ELSE IT-SUBJ-THAT IT-SUBJ-INF
IT-SUBJ-FOR-NP-INF (MEANS *GENERALITY *MATERIALITY)))
(IN (PREP (MEANS *RELATION *INTERIORITY *ENTRANCE
*ROUTE *MEANS)))
(THE (DET SG PL MASS DEF))
(PERSIAN (TOKEN (SHAPE CAP) (NAMES PERSIAN_GULF)))
(GULF (TOKEN (SHAPE CAP) (NAMES PERSIAN_GULF)))
(FOR (PREP
(MEANS *TIME *SUBSTITUTION *ATTRIBUTION *DISTANCE
*INTENTION *DEPUTY *AID *RECEIVING)))
(WHICH (PRO SG PL NEUT REL DEF (MEANS *CHOICE)))
(IT (PRO NOM OBJ SG 3P NEUT DEF))
(WAS (AUX PAST SG (ROOT BE)))
(NEVER (ADV NO-AUX-PRE-MAIN PRE-AUX POST-AUX-PRE-MAIN
INTER-AUX NEG (MEANS *TIMELESSNESS *NEGATION)))
(REALLY (ADV ADJ-SPEC ADV-MOD NO-AUX-PRE-MAIN PRE-AUX
POST-AUX-PRE-MAIN INTER-AUX SENT-SPEC
(MEANS *EXISTENCE *GREATNESS *TRUTH)))
(DESIGNED (VERB PAST PASTP TRANS INTRANS IO-NP-C NP-AS-NP-C
NP-INF-C (ROOT DESIGN) (MEANS *INTENTION *PLAN)))
(<~PERIOD?~> (PUNC))
(IT (PRO NOM OBJ SG 3P NEUT DEF))
(IS (AUX 3P-SG (ROOT BE)))
(DESIGNED (VERB PAST PASTP TRANS INTRANS IO-NP-C NP-AS-NP-C
NP-INF-C ROOT DESIGN) (MEANS *INTENTION *PLAN)))

...

Figure 5: Sample Data from our SCANNER (filtered)

4.6. Define Example MEANINGS in the CFT and Extension Lexicons

[Budget: focused on meeting Goal 3.2, this will drive about 18+9 man weeks by PI]

One key goal of our research is showing that complex situations of the sort described in English paragraphs can be straightforwardly represented by extending the modeling technology of Topic Maps to the situation models of case frames. But a second goal – specific to NLP applications – is finding the correct templates from the MEANINGS of head-word TERMS in our CFT (which gets shadowed by any extension lexicons the end-user chooses to add).

How does this lookup process work? The semantic markers used in our current lexicon (see “MEANS” tags in figure 5) are actually names for the 1,000+ semantic categories defined in *Roget's Thesaurus* [16]. Technically, they cite the categories in that book which would contain the related root word if low-utility(oddball) cases get dropped. (They were indeed culled by our linguist-editor staff when Lexikos hand-classified all root words in our lexicon.).

Such features are NOT used yet by our parser or grammar. They were added to support a future *Context* model, which MODELER can now arrange. This will test our hypothesis – backed up by a non-trivial editorial investment - that such markers are indeed a very valuable resource for any computing system trying to assign meanings to English words, phrases, or clauses.

The scope of this proposal precludes us showing all details of using such Roget's data, but the core principle is easy to explain. Any thesaurus, including Roget's and our own related CFT, use a tree-like classification system which can be exploited for data inheritance. To add data logically into 1,000+ categories of words, therefore, the tree of Roget's categories lets us actually store (and enter) it only in the top few dozen or so – the most general MEANINGS – then let the others inherit it logically at run time, or else explicitly override it, again at the most general level possible.

With an indexing system for roles, such approaches can save us a lot of boring data entry work, and make our semantic data more easily maintained and edited. Our idealized English ontology adds adequate notation to internally classify and model MEANINGS, then find them at need using *data inheritance mechanisms*. The basic mechanisms will be defined and supported in task 4.3. Task 4.6 then lets us test and verify or refine them, as we build up all those basic concepts listed in 3.2

4.7. Our Phase I Research Report and Proposal for Phase II

Results from all sub-tasks will be included in a final report, which documents our prototypes, and cites them as props to help explain what we learned, and will try to do next. We regard this document as important, and we will give it the proper amount of attention, including help from a technical writer. [Budget: This consultant sub-contract will help meet Goals 3.2 and 3.4]

In past R&D work on NLP, Lexikos has often organized such a document into short technical appendices on distinct subjects, related and summarized by a top level overview. On this project we expect to do the same, as such appendices can be successfully repurposed for training and proposals on Phase II, and help provide a base of documentation for our overall system.

5. Related R/R&D

Shortly after forming Lexikos in 1986, the PI and other refugees from Wang's Directed Research group (on using NLP/AI for Office Automation) began to assemble a large and accurate base of lexical data designed for use by English parsers. It was actually this lexical data – see figure 5 above - rather than the grammar or our parser's analysis logic, which occupied most of our time. It also took the bulk of the \$800,000 invested by 1991 into our overall parsing system.

5.1 *Our Semi-Deterministic English Parsing System*

Details on this successful 4 year effort are available in our final SBIR report [1], delivered with software to staff at Wright-Patterson AFB at a final presentation by the PI in October 90.

To make it work, we had to expand the deterministic analysis framework Marcus made famous with his 1980 thesis [29], and add limited backtracking. Below in figure 6 is its human-readable output, showing the sort of phrases (NP) and clauses (S) we now plan to map into linked topics. Nearby linguistic features, combined with our semantic markers, make this straightforward

5.2 *Continuing R&D on Topic Maps and CONTEXT*

Lexikos explored the concepts outlined in this proposal on several prior occasions. In 1990, we started to encode them in the Common Lisp environment. Those plans were interrupted until the mid '90s by a dismal economic climate, then delayed once again when Netscape and then Java showed that PC computing platforms would soon undergo fundamental changes.

By '01, Common Lisp R&D seemed near death, but RDF [4] had appeared for semantic models (or so it seemed). Under it, our elements of CONTEXT would become rules in description logic, inferred by HP's Jena [14]. But confusion persisted over OWL specs [11], coding tools, and redefining our WORDS ontology. So design ideas grew, but without a delivery vehicle

That changed in '02 when we noticed that Topic Maps [21] offered us an XML metadata format far more accessible for immediate R&D. With a small but established user community (mostly in Europe), far lower learning curves, and a powerful paradigm for modeling, TMs use Frame-like associations [25] not logic. The (relieved) PI joined their ISO standards group.

In some ways equivalent to RDF (two-way mappings exist at run time [24]), a standard Topic Map ("TM") would compete well against the first 80% of any well crafted RDF ontology we might have defined for CONTEXT. So we only needed to select it, then add the last 20% - early drafts of which are now published [5] using the TM-standard ("PSI") methodology.

This R&D continues part time while we seek funding from various sponsors for combining Topic Maps with NLP, and also get to know (by email) the heads of related engine vendors and open source teams. They express support for our plans, but are too small to fund them

	S		1 (S DECL MAJOR)
	NP		2 (THAT-C SG DET NP DEF 3P)
THE	"THE"		0 (DET SG MASS PL DEF NGSTART ART)
	NBAR		5 (NBAR SG THAT-C)
REPORT	"REPORT"		3 (NGSTART NOUN SG NONCOUNT THAT-C)
	S		12 (S SEC COMP-S THAT-S)
	COMP		11 (COMP THAT-COMP)
THAT	"THAT"		4 (COMP THAT-COMP)
	NP		7 (PRON-NP NP PERS 3P SG NOM)
SHE	"SHE"		6 (NGSTART PRONOUN PERS F 3P SG NOM)
	AUX		13 (PRES AUX PERF)
HAD	"HAVE"		8 (VERB AUXVERB PAST PASTP TRANS NULL-INF-C NP-VP-C S-PASTP-C NP-VING-C NP-ADJ-C)
	VP		19 (VP)
WRITTEN	"WRITE"		9 (VERB PASTP TRANS INTRANS THAT THAT-C NP-THAT-C IO-NP-C Q-C)
	AUX		20 (AUX PAST)
	VP		21 (VP COPULA)
WAS	"BE"		10 (VERB AUXVERB PAST SG COPULA COP-NP ADJ-C COP-ADV COP-PP NULL-INF-C Q-C FOR-NP-INF-C THAT-C PAST-C)
	AP		15 (IT-SUBJ-FOR-NP-INF IT-SUBJ-INF AP NONDET-AP ADJ-COMP)
INCORRECT	"INCORRECT"		14 (IT-SUBJ-INF IT-SUBJ-FOR-NP-INF)
<~PERIOD~>	"<~PERIOD~>"		16 (FINALPUNC DECL)

Do you accept this parse as valid? (Y or N) No

	S		1 (S DECL MAJOR)
	NP		2 (3P DEF NP DET SG THAT-C MODIFIED)
THE	"THE"		0 (DET SG MASS PL DEF NGSTART ARTT)
	NBAR		5 (NBAR SG THAT-C)
REPORT	"REPORT"		3 (NGSTART NOUN SG NONCOUNT THAT THAT-C)
	S		23 (S SEC RELATIVE)
(NP-2)	COMP		22 (SG 3P COMP WH-COMP REL UTILIZED)
THAT	"THAT"		4 (WH REL UTILIZED)
	NP		7 (PRON-NP NP PERS 3P SG NOM)
SHE	"SHE"		6 (NGSTART PRONOUN PERS F 3P SG NOM)
	AUX		24 (PRES AUX PERF)
HAD	"HAVE"		8 (VERB AUXVERB PAST PASTP TRANS NULL-INF-C NP-VP-C S-PASTP-C NP-VING-C NP-ADJ-C)
	VP		26 (VP)
WRITTEN	"WRITE"		9 (VERB PASTP TRANSINTRANS THAT-C NP-THAT-C IO-NP-C Q-C NP-INF-C)
(COMP-22)	NP		27 (NP NOT-MODIFIABLE NTRACE)
	AUX		28 (AUX PAST)
	VP		29 (VP COPULA)
WAS	"BE"		10 (VERB AUXVERB PAST SG COPULA COP-NP ADJ-C COP-ADV COP-PP NULL-INF-C Q-C FOR-NP-INF-C THAT-C PAST-C)
	AP		25 (IT-SUBJ-INF IT-SUBJ-FOR-NP-INF AP NONDET-AP ADJ-COMP)
INCORRECT	"INCORRECT"		14 (ADJ IT-SUBJ-FOR-NP-INF IT-SUBJ-INF UTILIZED)
<~PERIOD~>	"<~PERIOD~>"		16 (FINALPUNC DECL)

Figure 6: Two Parse Trees for a Syntactically Ambiguous Sentence

6. Relationship to Future R&D

In broad strokes, the work needed later falls into three parts. Major examples of efforts in the last two areas are discussed further below in sub-sections

- Integrate, refine, repackage, document, test, and our release text analysis software. This is mostly routine software engineering on Java Web apps, necessary but not novel
- Optimize CONTEXT for a **specific** application and/or corpus of the sponsor, by adding focused extension lexicons, plus follow-up queries of the Topics cited in *charts*
- Refine/expand our semantic data on word MEANINGS, and work with standards groups to popularize our linguistic models and ensure they stay valid XTM (and/or RDF).

Lexikos expects the *Agency* will take several people 2-3 years to develop into a self-funding site. Once phase I lets us demo prototypes on-line, we will seek new partners at the business level, both technical and financial. Our final R&D plans must emerge with their aid and advice.

6.1. Phase II – Optimize CONTEXT for Application-specific Knowledge Bases

MODELER used in Standalone Mode (no parser, just WORDS) lets us build up Context as a knowledge base of Topic types and constraints. The ontology thus created will aid parsing decisions, by constraining word and phrase MEANINGS to match expected forms.

Once SCANNER and PARSER get integrated, Training Mode will become a big focus, as it tests these constraints. It also helps us build application-specific extension lexicons, which expand on CONTEXT in several ways *before* similar input text is read in later sessions.

A wide range of specific applications might be used to focus such training mode II work. Ideal candidates will be characterized by a few specific types of short documents in simple textual formats, with well focused specialty vocabulary, being actively input by their author or by a specialist-clerk who can interact with our software as the input text is scanned and transcribed.

Intelligence summaries fit the bill perfectly, especially careful descriptions of images. Unclassified equivalents could be abstracts of publications, lead paragraphs in news stories, or even emails. In all cases, the basic subject matter of the text must be known in advance, so the appropriate semantic lexicons can be assembled, pre-loaded into *CONTEXT*, enlarged as new vocabulary appears, and updated in usage to manage statistics on TERMS and MEANINGS actually encountered. Conceptually, vocabulary limits and statistics serve to help filter out unlikely words senses, just as in statistical NLP parsing methods, LSA, etc.

Direct queries of *CONTEXT* can be initially used to help test system understanding, and they can be done in several ways, including Scheme or Java methods. Currently the vendors of Topic Maps are competing for the most effective SQL-like or Prolog-like languages, and requirements for an ISO-standard "TMQL" query language now exist [20]. One query style we will explore in phase II, limited to interactive dialogs, is having a training-mode user phrase questions as statements, but substitute for the desired answer some question word such as *where, when, who, why, or how*.

In phase II, we will also develop automatic mode by asking DIALOG code to *heuristically* answer the questions it asks a training-mode user, then to log both the user's answers and its guesses. With offline comparisons, they give us a quantitative way to measure (and gradually lower) error rates. Automatic mode will work best in applications where modest error rates were acceptable, such as:

- Text mining or web page spidering where significant errors are already anticipated
- Large scale text transcription for information retrieval or a pre-pass at its translation
- Bulk transcription of text or voice exchanges in training mode (versus real operations)
- Voice synthesis improvements, using text understanding to aid word pronunciation

Voice recognition improvements also seem possible, but here a vendor must tightly couple his recognizer with our SCANNER at the level of input words and phonemes, not full paragraphs. The feedback loop at the right of the page-4 NLP process diagram symbolizes this effect.

6.2. Standards - Using Roget's as a Neutral "Book Code" for Upper Ontology Concepts

Roget's Thesaurus has a long history of refinement, a huge base of user acceptance, and a vast amount of work time by its *seven generations* of human editors. Under any reasonable estimate, the editorial investment in it rivals that for OpenCyc [18] which claims 650 man-years!

These works are quite different, yet they can be effectively combined. Both deal with human linguistic concepts and both create 6,000 or so nodes in a tree that can be used data inheritance. But each of the works below has *different semantics*, so they complement more than compete:

- *Roget's* "categories" average 250 or so root words and phrases of related meanings. Each of 1,000+ sets is then clustered into sub-sets of nearly synonymous terms and idioms.
- OpenCyc models each concept by using team-assembled formal logic, then adds maybe another 20 assertions on average to add common sense knowledge about each concept
- SUMO [33] –Similar but smaller, it has standards support from IEEE, whose members have integrated WordNet and other selected ontologies from AI and KR experts .
- FrameNet [3] and PropBase [32] focus specifically on predicate/semantic models

Each of these works is important in its own way, and it gives a different view of one English vocabulary, constantly shifting. On-line collections are proliferating fast, and they extend innumerable printed dictionaries, glossaries, and lexicons, many of which will follow.

It seems obvious such sources must converge on the Internet, likely as a networked lattice, rich with cross-links, but persistent in all their individual identities and advantages. *Roget's* can help this lattice to coalesce sooner, by serving as a neutral global index of 1,000+ major subjects, under which the competing technical models of each concept can be found and contrasted .

Our lexicon already uses those categories, so it can find any concept models we might want to attach to them. If our users register additional concept models, it will easily find them too, as extensions or replacements. During phase II, we will refine this phenomenon using extension lexicons, and encourage other ontology makers to provide cross links in bulk. In this way, we can make our own lexicons more powerful, and MODELER more broadly applicable.

7. Commercial Applications Potential

Lexikos will initially reuse the same sales strategy we did in 1986-90 – license our software tools to larger organizations. That plan was working fine until economic declines blindsided us, and many changes since only make the strategy now seem easier to pursue. They include:

- Platforms:** 14 years of Moore's law, the Internet, and portable J2EE tool sets
- Resources:** FrameNet and Open Source coding tools facilitate web-based NLP work
- Standards:** Instead of ad hoc metadata, now we XML, RDF, XTM, OWL
- Partnering:** Pre-web, nearness mattered. Today sales can be made worldwide

The commercial Internet platform, with distributed Java, metadata, web services and B2B sites is a whole different world from 1990 workstations. But at its core, NLP products still run on good lexical data, and that is still expensive to assemble, because much of the work must still be done manually - one word or meaning at a time. If we can help clients here, with bulk data or tools or some combination, we will find customers within our traditional NLP tools market. Today, it has expanded to include tools for text mining, voice processing, and call-center systems

Increasingly, a separate market seems emerging in knowledge representation, which today is focused mostly on metadata. If MODELER can help its user describe something in a formal way, which lets that same thing be found later by someone else who needs it, then it will find a market even if the metadata used is not proper English. Content and Knowledge management tools for organizations exemplify this idea, as do the worlds of UDDI [9] and ebXML [10].

Topic maps already play roles in these areas, and we hope to find early customers in them too. Trying to build relationships, we have already visited large-scale TM users, such as the Y12 lab of DOE, whose TMs help to bulk-classify sensitive documents on weapons technology. Nearby Oak Ridge National Laboratory also expressed interest in working as partners with us on a Phase II effort, even offering to help us actively write that proposal at the proper time.

We do not see ourselves so much in the NLP applications business as in the NLP tools business. As partners on contracts with application-development firms, we can supply many government agencies, corporations, and other large users of text, and help meet the huge emerging need for IA-like NLP technology which turns up again and again. Potential R&D partners that I have met seem more than happy to help deliver it, once we show persuasively that it works.

Demos and documentation are vital to making such budding relationships pay off, and there is no substitute for them. If you help us get these off the ground with phase I support, I am confident that Lexikos could finance further work on MODELER from consulting and R&D contracts, and attract open source workers on expanding the semantics models and coding tools.

Intellectual property gets tricky, but we can handle that with site licenses. And on technical levels the design of the *Agency* is capable of keeping any proprietary data safely on our server, and downloading to each client only those portions needed for specific tasks. No plan is truly failsafe, but this plan, routine security measures and licenses would impede most competitors.

8. Key Personnel & Bibliography

Dan Corwin, PI and Lexikos CTO

For me, the *Agency* architecture has been a long-term personal effort, started in 1972 at MIT's Draper Labs where I worked writing space-vehicle simulations (Skylab, Apollo-Soyuz). For their upcoming Space Shuttle simulator, NASA wanted new software that could read, validate and index in-line code documentation. That seemed a good PhD thesis topic for me to pursue in the MIT AI Lab, where I was a part-time grad student. Terry Winograd, my TA, had just built his SHRDLU system [19], and I resolved to turn his work into a practical software tool.

My career has since let me try, not in academia but as a commercial software developer. I wrote three major WP systems for Wang Labs and Lotus, plus 6-8 text-analysis routers, editors, and news filters as small products, all involving some custom scripts and interpreter. Their designs have impacted *Agency* architecture more than anything new I might offer in NLP theory per se.

For seven years after my first WP work I headed Wang's "Directed Research" team in office automation, as an architect leading a team of 6-12 good developers. In '80 to '86, we competed with IBM's "Epistle team" trying to put a proofreader into a WP editor (like MS-Word's "Auto-correct", but 15 years sooner using 1% the memory). We also put several dictionaries into a dozen then-unique "Author Aids" that extended WP with scripted document assembly logic.

In '86 I spun off Lexikos, taking along a small but select team. We got early funds from GE R&D Labs, who paid us \$150k for the extension lexicons they used in [7], a financial news analysis program. SBIR also helped - two phase-I awards, then a \$563,000 phase-II with the Air Force [1], which *succeeded* at least through syntax analysis (our first focus). DEC, Apple, and ADS (a CIA tools shop) - declared themselves eager to help us add semantics in phase III sponsored R&D. But as part 5.2 explains, a rapid slide in the economy and other events long delayed this.

Active as a scripting and editor consultant during much of the '90s, I shifted into Java in '97. But language work remain my first love, so in 2000 when I moved to Maine, I reopened Lexikos to combine that interest, scripting, and semantic modeling part time while consulting on Java as my day job. My recent publications are on-line, and include [5], [25], [27], and [30].

William R Corwin, Lexikos President

As a Program Marketing Director at Oak Ridge National Labs, with a technical MIT education and two dozen years running advanced Lab research programs in metallurgy, Bill has an intuitive understanding of what it takes to make a high tech research program work. Although he is not a computational linguist, he is a technologist of the first order, and understands well the complex process of blazing new technical trails in a research environment

Bill actively helps to promote our technology, especially within government circles, and has guided our initial steps toward marketing and partnership formation for the company.

9. Company Information and Facilities

Lexikos Corporation reopened in 2000 for consulting and linguistics software tools. Our vision and mission today, as at our 1986 founding, is to build a stable team of experts who know and love the ways words and language convey information and can translate that knowledge into world-class software simulating the process, in modules which can be tested, explained and improved. The driving force is cognitive and linguistic science, and solving challenging puzzles.

We use PCs for most development. For posting early reports and *Agency* demos, we will use a staging web site Lexikos maintains at ShadowCats.com [8] (a sales partner), and/or at Lexikos.com, and/or at the open-source CVS repository at sourceForge.net which the PI helps to manage as co-manager of the Maine Java User group, and/or at a similar site [14] for our XTM engine, TM4J.

10. Consultants and Sub-Contracts

Below are summaries of two prototype elements slightly off the main R&D path for *MODELER*, which would benefit from special talents and skills. For them, if we get a phase I award, Lexikos will hire local contract coders. We have multiple options for both positions, and at need the PI can fill in on either one, so no firm names are yet attached. We also need help on tech writing.

10.1. User Interfaces for DIALOG and Subsequent Agency IA Clients

This sub-contract focuses on goal 3.3 and task 4.4 - the intuitive, mixed mode IA display shown in Figure 3. It should be created by a UI specialist in Java, and later on, in .NET client code. A temporary developer who knows both platforms and has artistic talent would be ideal.

10.2. SCANNER and TRANSCRIBER Upgrades in Common Lisp

This sub-contract is summarized by goal 3.4. Task 4.5 is part of it. It essentially involves a port of our existing code or its data into a web environment, after an analysis of the source code and current Lisp systems to decide the best long-term engineering path for these two modules. Such analysis and redesign work would be sufficiently OS-sensitive and specialized to Common Lisp to make me hesitate about doing it myself, so another coding specialist seems recommended.

A new Common Lisp or Scheme package with a compiler to Java would be one option, or a tool such as CL-HTTP. A simple (temporary?) alternative is as a background server task, interfaced to Java servlets by input and output queues. Another path is a direct port of source code into JScheme or Java itself. This would be more practical for *SCANNER*, as it is a simpler code module. Our grammar and parser are separate Lisp modules, and might get different solutions.

10.3 Technical Writer

For help on documenting our system, and completing our final report, Lexikos will also seek help from an hourly tech writer, so the PI can focus more fully on software development

11. Related Proposals to other Federal Agencies

Original SBIR phase II work on syntax analysis was successfully concluded in October 1990, and reported to the US Air Force [1]. We restarted work on semantics three years ago. After encouraging feedback on submissions last year, we now try again with similar proposals [2].

References – also on-line at <http://www.lexikos.com/metadata/links.htm>

- [1] "Representing Knowledge in Idealized English", Lexikos SBIR Ph II, 1990
- [2a] "Charting Paragraph Topics", Navy '04 SBIR Ph 1 proposal on Topic N04-119
- [2b] "Charting Paragraph Topics", USAF '04 SBIR Ph 1 proposal on Topic AF04-097
- [3] <http://citeseer.nj.nec.com/context/33135/0> on "The Case for Case" C. Fillmore, 1968
+ <http://www.icsi.berkeley.edu/~framenet/> - FrameNet: collected case frame examples
- [4] <http://www.w3.org/RDF/> - '99 metadata recommendation by W3C
- [5] <http://www.lexikos.com/psi/words/> – online specs, for Idealized English Ontology
- [6] <http://jscheme.sourceforge.net/jscheme/mainwebpage.html> - Jscheme - Lisp freeware
- [7] "SCISOR: Extracting Information from Online News" Jacobs & Rau; GE R&D Labs
- [8] <http://www.shadowcats.com> - sales partner's planned staging area for *The Agency*
- [9] <http://www.uddi.org/about.html> - new IBM/Microsoft registry for B2B web services
- [10] <http://www.ebxml.org/> - similar new national registry emerging from OASIS
- [11] <http://www.w3.org/2001/sw/WebOnt/> - OWL working group home
- [12] *Artificial Intelligence - A Modern Approach* - Russel & Norvig, Prentice Hall 1995
- [13] <http://www.hpl.hp.com/semweb/doc/tutorial/index.html> - Jena Tutorial, HP team, 4/02
- [14] <http://sourceforge.net/projects/tm4j> - TM4J homepage - Open Source Topic Map engine
- [15] <http://java.sun.com/j2ee/> - J2EE home page at Sun Microsystems
- [16] *Roget's Thesaurus*, Fourth Edition - Harper & Row
- [17] *Natural Language Understanding* - James Allen, Benj Cummings, 1987 & 1995
- [18] <http://www.opencyc.org/> - OpenCYC HomePage
- [19] "Understanding Natural Language" - T. Winograd; *Cognitive Psychology*, 1972
- [20] <http://www.topicmap.com/> - a portal for the commercial Topic Map community
- [21] <http://www.topicmaps.org/xtm/1.0/> - ISO Standard 13250 on Topic Maps
- [22] "Attention, Intention & Structure of Discourse" - Grosz & Sidner - *Comp. Linguistics*, '86
- [23] <http://java.sun.com/products/jsp/jstl/> - Java Standard Tag Library
- [24] <http://www.semanticwebserver.com/publications.html> - Integr'n of TMs & RDF. Moore
- [25] <http://www.isotopicmaps.org/pipermail/tmcl-wg/2003-March/000021.html> - Frames
- [26] <http://www.topicmaps.org/xtm/1.0/#elt-association> – in [21], but specific to associations
- [27] http://www.lexikos.com/metadata/ofw_notes.jsp – PI Trip Report on 1/02 OFW Conf
- [28] <http://www ldc.upenn.edu/> – Linguistic Data Consortium – for simpler lexicons & corpii
- [29] <http://citeseer.ist.psu.edu/context/84907/0> – *Theory of Syntactic Recognition*, Marcus, 80
+ <http://www.cis.upenn.edu/~mitch/> - his homepage and recent work
- [30] <http://www.infloom.com/pipermail/topicmapmail/2003q3/005014.html> – schema thread
- [31] <http://www.w3.org/TR/xmlschema-2/#built-in-datatypes> – Datatypes in XML Schema
- [32] <http://www.cis.upenn.edu/~ace/> – PropBank project – Martha Palmer & Mitch Marcus (PIs)
- [33] <http://ontology.teknowledge.com/> – SUMO – IEEE SUO Working Group
- [34] <http://www.adlnet.org/index.cfm?fuseaction=scormabt> – ADL Overview of SCORM