SIDECAR: Semantically Informed, Dynamic Engineering of Capabilities and Requirements

Phase 1 Report
November 18, 2010

Douglas Lenat, Ph.D, Cycorp
Benjamin Rode, Ph.D, Cycorp
Nathan Winant, Cycorp
Robert Kahlert, Cycorp
David Baxter, Ph.D, Cycorp
Steven Collins, Cycorp
David Mayo, Everware
Gary Berg-Cross, Ph.D, Everware
Chris Gunderson, Naval Postgraduate School

Technical Contact: Lenat@cyc.com 512-342-4001
USG POC: Lt. Col. David J. Yost david.j.yost@usmc.mil

Table of Contents

1 Executive Summary

Motivation. There are serious size, speed, and security (S^3) problems in the Defense Community fielding a cohesive Global Information Grid, and in the Acquisitions and Requirements Communities producing and maintaining up to date Capability Development Document (CDD) and Initial Capability Document (ICD) models of each program individually let alone their interdependencies. [Gunderson, 2010] Those problems can't be avoided, since effective shared information access is already a key factor in warfare, and Command, Control, Communications, and Intelligence (C3I) is already heavily dependent on computer networks.

But looming just *beyond* S^3 is an even higher peak: *cognitive capability,* the bottleneck caused by the lack of semantic depth of understanding with which our software processes information. That lack of semantic depth manifests, e.g., in the brittle way DODAF (Dept. of Defense Architecture Framework) 2.0 formats display DoD architecture products and the voluminous amount of statutory, regulatory, and prescriptive policies and directives emanating out of all DOD entities from engaged MAGTFs up through the Offices of the Secretary of Defense. Maintaining a single source of truth when hundreds of people are changing individual overlapping elements is less than straightforward: a change made by one individual can have consequences for many others in different parts of the enterprise and in related enterprises, and no mechanism exists for highlighting or resolving ambiguity. [Hopkins & Jenkins, 2008, p. 129] It would be absurd to imagine the intelligence community employing human idiot savants in all the crucial roles of (everything except the very last stage of) processing information up through high-level predictive analysis, and yet we blithely rely on brittle *software* idiot savants in exactly those crucial roles! What is needed is an automated system that is capable of combining and interlinking different representations within a single repository containing the aggregated knowledge of the existing operational environment, in a way which supports a documented baseline of understanding. [Hopkins and Jenkins 2008, p. 166]

Thanks to recent progress in automated reasoning, we can now do something about this.

The task: A program serving as an intelligent auxiliary to the requirements officer assigned to produce a Capability Development Document (CDD) for a target system, say DCGS-MC. This task is reminiscent of filling out a complex income tax return (e.g., having to cope with legacy and changing regulations, policies, and directives), but in some ways much more complicated because the to-be-developed target system must be properly linked to all other relevant existing information/communications system architectures as represented in their requirements documentation (e.g. CDDs and CPDs).

SIDECAR is a pilot project performed in 2010, under USMC direction, to demonstrate this capability. The contractor, Cycorp (www.cyc.com), was able to execute this as a small ($300k) effort by leveraging its enormous already-existing ontology, knowledge base, natural language dialogue and inference engine technologies – collectively called Cyc – extending each of these components modestly only as needed. SIDECAR starts with partial information about relevant systems such as the Marine Corps Intelligence, Surveillance, and

Reconnaissance Enterprise (MCISR-E) and organizations such as the Joint Requirements Oversight Council (JROC), and a model of what information goes into which sections of a CDD (i.e., a declarative representation of which set of questions' answers *comprise* each section of a capability development document). SIDECAR uses that model to drive a mixed-initiative clarification dialogue, with its user, to obtain more and more information – much as TurboTax does in *its* domain. In this case, SIDECAR learns more and more about the target system, DCGS-MC. Having a *CDD model* means that all this new information is relevant to, and has a well-understood place in, the CDD. Although the dialogue is in English, the internal model in SIDECAR is represented in formal logic (n^{th}-order predicate calculus). Using Cyc's natural language generation capability, each of these assertions is converted into English, and SIDECAR's CDD model guides it in placing those English sentences in the correct order to generate the actual English CDD.[1] This happens in real time, incrementally, so the growing state of the English CDD can be viewed at any time. But instead of editing *it,* if something needs to be changed, the user is pointed back to whatever SIDECAR menu or question led to that particular English sentence or table entry. They modify their choice on that SIDECAR screen, and the English CDD updates.

SIDECAR partially *understands* what it's doing, in the sense that it can use its partially-completed model of the CDD-in-progress to infer such things as:

(1) Is there a contradiction between what the user just told me and everything else that's already known?
(2) Given what the user just told me, can I guess at any other questions' answers? If not, can I at least eliminate some of the possible answers to some of the questions? Are some *entire questions* now completely moot?
(3) At any given moment, where should the user best spend their effort, and focus their attention, next?
(4) If some of the alternative design choices have been left unspecified by the architects, what are the pros/cons of each alternative?

In the current pilot project version of SIDECAR, all 4 of these types of beneficial inference already clearly occur.

Future capabilities: Results to date indicate several useful directions in which this effort could be productively extended, deployed, and make an operational difference:

- Capability (4), above, could be expanded, producing a version that the software architects could use in designing the programs prior to the documentation phase.
- SIDECAR could be utilized in training new Acquisitions personnel.
- Regression testing against policy changes: rerun each program's design automatically every day, and auto-alert if new regulations – or changes in technology capabilities, vulnerabilities, or costs – or " real world" changes (e.g., political boundaries or conflicts) make some system's detailed design now incompatible with its intended capabilities.

[1] Instead of producing the final English CDD itself, SIDECAR could at this point hand the English sentences and paragraphs to an application such as CDTM to assemble and archive. Even in this case, the declarative logical SIDECAR model would be archived and associated with this CDD, much like a TurboTax **.tax** file, so a user could later load and resume editing that CDD using SIDECAR.

- Instead of just coping with those changes *after* they occur, SIDECAR could be used to answer "what if" questions of the models for any or all of these systems, to help analyze the impact of possible changes in policy, technology, etc. which are not yet decided.
- Represent existing systems and the terrain of regulations and policies in which they live. Those semantic models would jointly enable reasoning about FOS/SOS dependencies to be more complete and machine-reasonable, leading SIDECAR to make increasingly correct suggestions about what the new target system should link to (and how and why) or incorporate as a component, and suggestions about design alternatives' pros/cons.
- Extend that to interoperate with SIGINT, ELINT, HUMINT,… data represented in Cyc to detect attacks earlier and to generate populations of plausible threat scenarios which in turn could inform changes in policy, information collection, and operations.

Conclusion. Looking beyond raw speed, scale, and security bottlenecks, some action officers at Marine Corps Combat Development Command recognize the importance of developing a suite of machine-readable – and deeply machine-*reason*able – acquisition documentation which is deliberately focused on content rather than form, and which explicitly represents enterprise goals and the relation of program elements to those goals. SIDECAR demonstrates the feasibility of that vision, and exhibits several forms of TurboTax-like help that would be most welcome in practice.

2 Introduction

Phase 1 of the Automated JCIDS Project has been an approximately 1 person-year software development effort undertaken at Cycorp under the auspices of the Marine Corps Intelligence, Surveillance, and Reconnaissance Enterprise, during the period of time 22 February 2010 through 20 November 2010, under contract MTCSC 6006-270. Its goal is to demonstrate an effective capability to author, edit, interrelate, and analyze Capability Development Documents (CDDs) and (in later phases) ICDs, MUAs, CPDs, and related documents, for Joint Capabilities, Integration, and Development System (JCIDS) programs. (The JCIDS operational context w.r.t. SIDECAR is discussed in Sec. 3.2.2.)

A "proof of concept" demonstration system, SIDECAR, was produced, which interactively allows a user to carry out those three functions (authoring, editing, and interrelating) for portions of the DCGS-MC CDD. Much as TurboTax™ helps a taxpayer prepare a complicated annual IRS return, SIDECAR helps its user by guiding them through the numerous and complex items that comprise a CDD, watching every entry they make and pointing out contradictions (and helping the user resolve them) and automatically constraining still-valid answers to upcoming "blanks" the user will need to fill in. One special case of the latter constraining is when the number of choices drops to **0** and a question can be skipped entirely. Another special case is when the number of choices drops to **1**, and that remaining choice can be penciled in as the likely answer to that question.

To carry out this functionality, SIDECAR builds up an internal model of the state of the partially-completed CDD, and also a model of the user's interactions so far working with SIDECAR. As though SIDECAR had been used in the past, there is also a partial model of related CDDs, ICDs, CPDs, MUAs, IAVAs, programs, organizations, individuals, etc. (for acronyms, see Appendix A). Each of these models is a set of declarative assertions represented in a declarative Policy Markup Language (PML) we designed for this purpose, which in turn is a dialect of symbolic logic (more precisely, it is an extension of the CycL[2] higher order[3] logic representation language [Matuszek *et al* 2006]) where the constant terms of that logic [Enderton 2001] correspond to terms in the Cyc ontology [Lenat and Guha 1990]. That internal form is then used to drive the automated reasoning that SIDECAR needs to perform. An automated logic-to-English software module

---

[2] http://en.wikipedia.org/wiki/CycL

[3] "Higher order" means that the Policy Markup Language PML is more expressive than First Order Logic (FOL), which in turn is more expressive than Description Logics (such as OWL-DL) and similar Semantic Web markup languages (such as RDF). Higher order logics (HOL), also called n^{th}-order predicate calculus representations, allow one to state rules about rules, allow one to state various other metadata, and even to express nested modals and hypotheticals such as*: organization X wants to prevent agents of country Y from expecting that if network Z were used to carry out action A then agents of Y might be able to…* In short, anything expressible in English can be expressed in PML. Conversely, restricting our system to any representation language *less* expressive than PML would mean that there were some policies, contingencies, domain rules of thumb, etc. that could not be fully expressed in a way that an automated reasoner could use to derive the same conclusions, from a set of such assertions, that a human being would derive from them. In other words, using a higher order language equivalent in power to PML is both necessary and sufficient for the SIDECAR application to fully scale up and be deployed.

converts the statements in that model into English sentences which are then appropriately sequenced and formatted and together then *comprise* the final CDD itself. Keeping around a "state vector" of the SIDECAR dialogue is also worthwhile, much as TurboTax creates a persistent **.tax** file, so that the current state of the SIDECAR session could be loaded and resumed as desired at some future date.

The SIDECAR architecture (See Figure 1) consists in

- a Cyc Server, which is also where most of SIDECAR itself can be considered to reside and be running. The key subcomponents of the Cyc Server are
    - the Cyc Knowledge Base (KB) store, which contains half a million rules and assertions ranging from general common sense and world knowledge all the way down to JCIDS-specific domain knowledge, all represented in formal logic (the CycL[2] dialect of n^{th}-order predicate calculus).
    - the Cyc Inference Engine, which contains both a general theorem prover and over 1,000 specialized reasoning modules – many with their own redundant, idiosyncratic representations – which together are able to work effectively as a community of agents, to make the overall inference process orders of magnitude faster than it would be if one were to rely on the theorem prover alone.
    - a thin socket-run network stack to support API-level communication with Cyc applications such as SIDECAR.
- an Apache Tomcat server, which maintains a container of servlets, one for each simultaneous user.
- an internet browser on the end-user machine, which is talking with the Cyc Server through the Apache servlet.

Each SIDECAR user points their browser to a URL which causes the Apache server to spawn a new 'Servlet' in its Servlet Container, for that SIDECAR session with that user. Apache maintains a separate execution context, threads, etc., for each servlet. This Apache layer handles user authentication, security, information assurance, logging, restarting, billing, administration, etc. Besides the obvious benefits, this has the property that any one instance of any one such servlet crashing is very unlikely to crash any of the others. Apache was originally written in C, and Apache *Tomcat* is simply the Java version, and hence is extremely portable and stable. It can run on the same machine as the Cyc server or on a different server machine. If there eventually are a large number of simultaneous SIDECAR users, there could be multiple layers of many-to-many Apache Servers on different machines.

SIDECAR is an archetypical Web Application, meaning that the end user simply points their browser (such as Internet Explorer or Firefox) to a URL for the Apache server machine which then spawns a new servlet instance for that particular session, and talks to the Cyc server.

In addition to the SIDECAR application proper, our current architecture (and demo) also supports a Cyc-based query service called the Cyc "Analyst's Environment", or CAE. This was developed for the Cleveland Clinic, under an unrelated contract [Lenat *et al*

2010], to help clinical researchers formulate queries in their search for hypotheses and patterns in large amounts of patient data.[4] The analogous use here is to help SIDECAR users formulate queries about the system being documented (e.g., DCGS-MC) and the document itself (e.g., the CDD.) So, although CAE is not a *part* of SIDECAR, it is a useful "app" for the user to be running. The CAE client has a small disk and memory footprint, and is able to run on even low-end laptops and PCs. There are C and Java versions of CAE, and it currently uses a SWING-based GUI.

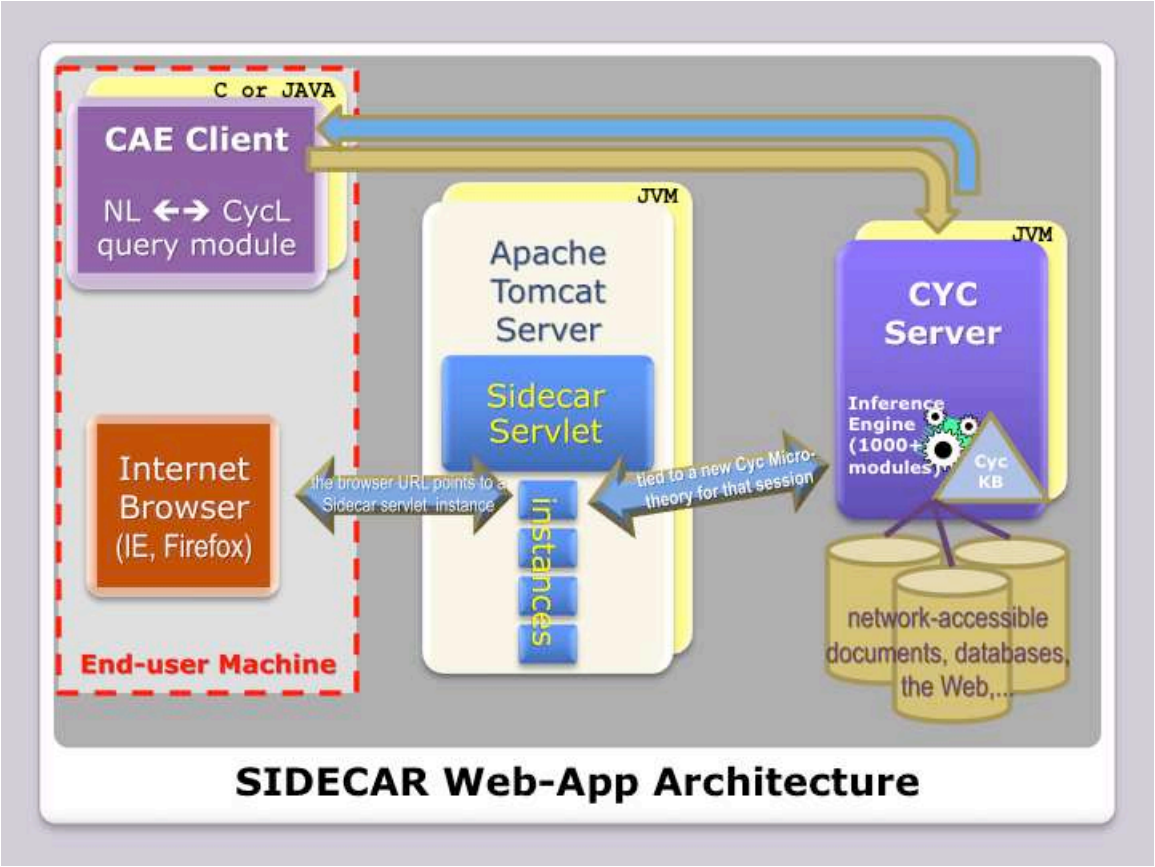


Figure 1. *The SIDECAR system architecture, including the Cyc KB Server and, for each simultaneous user, a SIDECAR Servlet taking with a lightweight client (browser). Also shown in this Figure is another Web-app, CAE, which is not part of SIDECAR but which a SIDECAR user can call on to help them formulate* ad hoc *queries to the growing KB model of the JCIDS system being documented (in this case, DCGS-MC) as the SIDECAR session proceeds.*

[4] CAE was originally developed for the US military, under a DARPA contract, and later for the US intelligence community, to help analysts formulate complex *ad hoc* queries against large amounts, types, and sources of (semi-)structured data.

The fundamental problem that SIDECAR is intended to address is the need on the acquisitions and requisitions side to model programs, development documents (ICDs, CDDs, CPDs, MUAs, etc.), policies, doctrines, combat theatre and geopolitical contexts, and technologies in a way that allows new development documents to be produced faster and with higher precision, and in a way that allows for active regression testing: flagging the user's attention when some policy or doctrine changes – or some technology changes – in a way that renders some elements of a document content now noncompliant, or more generally, when documents are produced that either make mutually inconsistent statements, or statements whose content implies possible adjustment on one side or the other.

One can be all too easily misled into thinking that the problem is primarily *quantitative*, relating to size, speed, and security of processing elements. While that is true to some extent, it is only part of the problem – in mountain-climbing parlance, those are false peaks. Focusing exclusively at the quantitative end masks a deeper, *qualitatitve* gap: specifically, lack of deep semantic understanding of the development context and document domain in the development tool suite. Today's databases and textual document preparation and processing software have the status of *idiot savants*: extremely efficient and effective with respect to formatting large volumes of data, and answering a narrow range of questions, but of no help in addressing the open-ended complex of issues regarding the interoperability of a proposed system's documentation with representations of policy, doctrine, and other systems.

SIDECAR is intended to address precisely this limitation in current document preparation technology. Its foundation is the Cyc® Knowledge Base, a dynamic, comprehensive ontology comprised of some 6 million assertions, built over the past 26 years at a cost of approximately $110 million and 1,050 person-years of effort.

The Cyc Knowledge Base represents information a formal logical language called CycL, which is accessible to machine reasoning using a dedicated theorem prover called the *Inference Engine*. The reasoning that is done is equivalent to resolution theorem proving[5] [Robinson 1965], but with efficient inference support provided by over 1000 special-purpose reasoners called *heuristic level (HL) modules*.[6] Each HL module supports some frequently-used reasoning pattern (e.g., graph-searching a transitive relation's "tree"), maintains a redundant data representation, encodes a special case reasoning algorithm that makes use of that specialized representation and that thereby enables Cyc to carry out that particular pattern/type of reasoning many orders of magnitude faster. So 99% of the time, Cyc HL modules are running, not the general Cyc theorem prover. For example, one HL module maintains a graph structure representation for each relation known to be *transitive,* so some class of queries that would otherwise have required an *n*-step proof can instead be discharged via a quick graph traversal search.

---

[5] http://www.cyc.com/doc/tut/ppoint/InferenceLogicalAspects_files/v3_document.htm

[6] http://www.cyc.com/doc/tut/ppoint/InferenceHeuristics_files/v3_document.htm

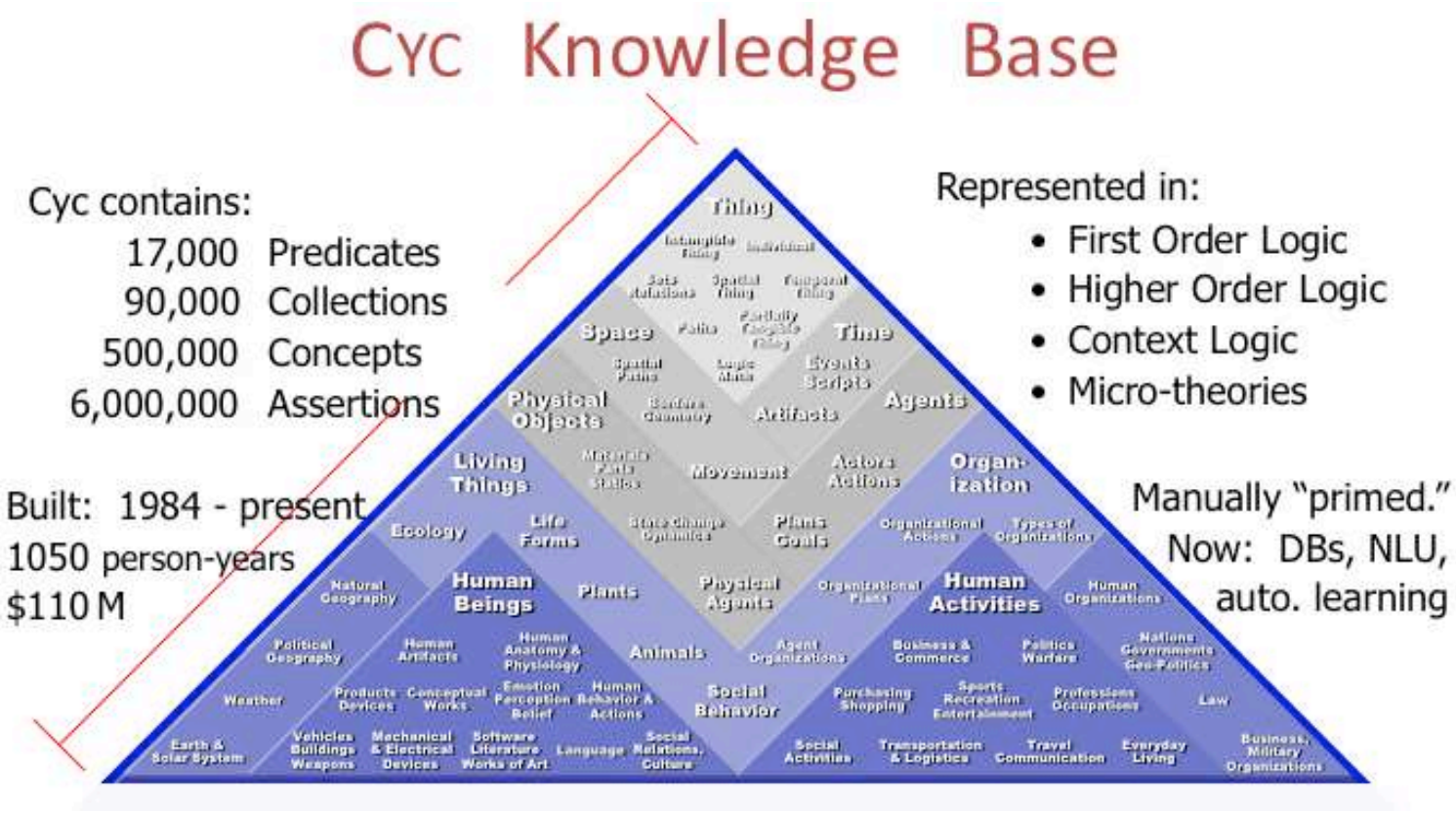


Figure 2. *A "50,000-foot view" of some of the topics covered by the upper and middle levels of the Cyc ontology. This diagram makes some significant simplifying assumptions: it does not reflect the fact that CycL admits faceting via higher-order collections; it does not capture the fact the knowledge (including inheritance relations) may be contextualized in relation to a context hierarchy, and it does not reflect the* ***multi-hierarchy*** *of the KB. Cyc's knowledge base (KB) is a set of millions of rules and statements in CycL, a formal language that is sufficiently expressive to capture all of the requisite background knowledge concerning the JCIDS domain, in a way that allows SIDECAR to proactively guide the user – and interactively work with them – in developing a JCIDS document based on understanding of the user's inputs.*

The motivation for using a formal language (such as symbolic logic) for representing the background domain and for handling the inferences in SIDECAR is threefold:

- First, semantics can be provided for selected fragments of the language such that truth-preserving rules of inference can be recursively defined and implemented through an automated theorem-prover (the Cyc Inference Engine), thereby enabling us to approach the problem of reasoning, inference, and justification within the domain with mathematical exactitude, and to provide formal recursive justifications for every result the system produces.
- Second, formalism admits introducing various forms of generalization (subset-superset links between 2 collections, but also specialization/generalization links

between 2 relations or even between 2 *contexts)* that allow us to succinctly capture knowledge in a way that scales efficiently to a broad range of use cases, while at the same time allowing for principled contextual control over the information that is accessed in a given inference (thereby excluding irrelevant information).

- Finally, use of formal representation supports integration with the Cyc ontology. This in turn is important for two reasons:
    - The breadth of the Cyc ontology is needed to avoid SIDECAR's behaving as an expert-system-like *idiot savant,*
    - The breadth and depth (content) of the Cyc knowledge base is needed to support the system's successfully carrying out two of the most important heuristic strategies that pervade our everyday life as human beings when we are confronted with novel situations:
        - falling back on increasingly general knowledge, and
        - analogizing to increasingly far-flung experiences.

The reason the Cyc Knowledge Base is referred to as a 'knowledge base' (as compared with a '*data*base') is that it stores information about classes and dependencies *between* classes, that can be used in turn to interpret and derive results from instance-level data that has itself been translated into CycL. Some of that information is straightforward simple subclass/superclass relationships which could be faithfully captured in (and used by) databases. However, a great deal of the information is not simple taxonomy, but takes the form of *rules*[7] which allow much more complicated reasoning to be performed.

Forward vs. Backward rules. The Cyc system distinguishes between rules which are so combinatorially explosive that they are only used in backward reasoning (solving sub-problems), versus rules which are so important that they are eagerly applied as soon as they become valid to apply in any circumstance. *Backward inference* is done at query time; more specifically, it is a query-sensitive search through possible proof space that is rooted in the query that is asked and bottoms out in true sentences that are directly provable from either the knowledge base or reasoning modules. *Forward inference*, in contrast, is performed eagerly at assert time among all the assertions in the KB that are asserted with direction 'forward' and continued to quiescence. Every assertion in the Cyc KB is labeled with a direction of either 'forward', 'backward', or 'code', usually at the discretion of the human Cyclist making the assertion. Only forward assertions participate in forward inference, while both forward and backward assertions participate in backward inference.[8] If an assertion is labeled 'code', then it participates in neither backward nor forward reasoning because some Heuristic Level module exists which carries out the

---

[7] Commonly occurring "schemas" of rules are represented by assertions using predicates with type-level arguments (which CycL allows) whose definitions can be expanded into rules. E.g., one such predicate is relationAllExists. The assertion (relationAllExists x y z) represents a rule that each y has a z playing the role x. For example (relationAllExists properPhysicalParts NuclearSubmarine NuclearReactor) expresses the rule that each nuclear submarine has, as one of its physical parts, a nuclear reactor; that rule is therefore written as what is syntactically a simple gaf (ground atomic formula) in CycL..

[8] Siegel, N., Goolsbey, K., Kahlert, R., Matthews, G. *The Cyc System: Notes on Architecture.* Whitepaper available at http://www.cyc.com/cyc/technology/whitepapers

logically equivalent form of obeying that rule, but vastly more efficiently than the Cyc theorem prover would be able to do.

Microtheories (Contexts). Information in the KB is contextualized to take account of the fact that things are often true in one context and less true, or simply false, in another context. E.g., things are true at one time and false at another; some rules of appropriate conduct apply at sporting events but not formal dinners (and vice versa); some things are true at one level of granularity or abstraction but not others; some things are known/believed by certain agencies or individuals and not known/believed by others. To keep locally-consistent but globally-inconsistent bundles of assertions from mixing too readily, Cyc overlays its knowledge base of assertions with a tapestry of regions of knowledge-space, sets of assertions sharing a common set of assumptions (about time, granularity, etc.) Each of these contexts is called a Microtheory and abbreviated Mt.[9]

CycL *queries* are specially formulated CycL expressions which can be run with associated inference parameters[10] in reasoning contexts to return answers, or *bindings*.[11] Users may pose a top-level query to Cyc, or an application (such as SIDECAR) may pose them, or Cyc itself may "broadcast" sub-sub-…-problems as queries (which Cyc recursively then works on). Here is a typical CycL query:

```
(thereExists ?AGENT
  (and
    (isa ?SYSTEM IntelligenceAnalysisAndProductionSystem)
    (hasAgents ?SYSTEM ?AGENT)
    (intendedTheaterOfOperations ?AGENT
      (TerritoryFn Armenia))))
```

Figure 3. *An open formula that can be used as a CycL query if asked in an inference context with appropriate inference parameters. "Open" means that the variable* ?SYSTEM *is unquantified; hence, each answer to this query will be some intelligence analysis and production system which is intended to operate in Armenia and hence satisfies the query. The listing of all those* ?SYSTEM *values returned by this query is therefore the set of answers to it.*

---

[9] http://www.cyc.com/cycdoc/course/contexts-basic-module.html

[10] *Static query parameters* are those that are fixed once an inference for that query is created, and cannot be changed if the inference is later continued, while *dynamic query parameters* can be extended after an inference is created, when that inference is later continued (i.e., when inference halts, it returns the reason why, and some reasons indicate that the inference could be continued with more resources). Many Cyc inference parameters lie along a completeness/efficiency trade-off from their most efficient setting to their most complete setting. Collectively, these parameters constitute a multidimensional parameter hyperspace from the most efficient extreme to the most complete extreme. In practice, a Cyc application developer may only need to know about, and use, a handful of parameter settings.

[11] http://www.cyc.com/doc/handbook/oe/14-using-cycl-queries.html

The query in Figure 3 asks for instances ("isa") of the Cyc collection called IntelligenceAnalysisAndProductionSystem such that said instances have agents ("hasAgents") intended to operate ("intendedTheaterOfOperations") in Armenia[12].

Notice that, in Figure 3, in addition to the items which appear in blue (CycL terms), the query contains two variables prefaced with question marks: ?SYSTEM and ?AGENT, which appear in black. The variable ?AGENT is said to be 'existentially bound' in this query context, because it appears inside of the existential expression thereExists. The variable ?SYSTEM, in contrast is said to be 'open' or 'free'. The idea is that the query is looking in the Knowledge Base (or in the proper subset of the Knowledge Base that is specified by the query context) for 'bindings' for ?SYSTEM – terms which, when substituted for ?SYSTEM in the query expression, will result in a true sentence.

It should be noted that the *semantics* of the Cyc relations that the reader sees – such as isa, hasAgents, and intendedTheaterOfOperations – are partially given by the definitions these terms have in the Knowledge Base (and they are all defined). The relation 'isa' is analogous to the 'element of' relationship in classical Set Theory. For example, the expression (isa Fido Dog) means that Fido is an instance of the class Dog. The expression (hasAgents ?ORG ?AGENT) means that ?AGENT is capable of (and tasked with) acting on behalf of ?ORG. And (intendedTheaterOfOperations ?UNIT ?THTR) means that ?THTR is a geographical region where the military unit ?UNIT is expected to be able to operate (it may not be the only, or largest, such place where ?UNIT may operate; e.g., if I ask Cyc a query for agencies which can operate in Texas, one of the answers will be the FBI and one will be the TexasNationalGuard, the former operating anywhere in the US and the latter generally restricted to operating within Texas.)

In searching for answers – bindings of ?SYSTEM – for the Figure 3 query, the Cyc inference engine has available to it not just facts which are explicitly present in the reasoning context, but facts which can be *derived* from other facts by the Inference Engine, using the aforementioned rules and rule-abbreviating assertions.

For example, suppose that Cyc has been supplied with the information that, within the Marine Corps Distributed Common Ground-Surface System (the DCGS-MC described below), there exists a military unit that has the entire Caucasus region as its intended theater of operations. This is an actual example of an assertion which the user tells SIDECAR, by the way, but in a simple, natural interaction. The SIDECAR end user is unaware, as they do that, that some formal CycL assertion like the following one (in Figure 3b) is being written by SIDECAR, automatically, and entered into Cyc's KB:

---

[12] The expression, (TerritoryFn Armenia), is what is known in CycL as a non-atomic reified term or NART: a term which has been composed using other terms. Because definitional facts about such terms can be automatically concluded from rules used in defining terms like TerritoryFn, NARTs can be a cost-saving option when it comes to representing many definitionally similar or analogous terms. It can also be expedient for *Cyc* to use function for creating NARTs that represent things it has just learned about. In this case, what is being referred to is the geophysical land (and airspace) of Armenia, not, e.g., the government of Armenia.

```
(thereExists ?AGENT
 (and
   (hasAgents
     (TextKAObjectFn
      (CDDSubSectionOfSubSectionTypeFn
       (CDDSectionOfSectionTypeFn  TestCDDDocument02  CDDIntroduction-PCW)
       CDDIntroductionSubSection-TitlePage-PCW)
      (KATemplateFieldOfTextTypeFn
        CDDIntroductionSubSection-TitlePage-PCW
        DODAgentAndInstrumentSystem  1)
     "Marine Corps Distributed Common Ground Surface System")
   ?AGENT)
  (intendedTheaterOfOperations ?AGENT (TerritoryFn Caucasus)))
```

Figure 3b. This assertion says that DCGS-MC is intended to have agents capable of operating in the Caucasus. It is produced automatically by SIDECAR, as a side effect of what seems to the end user like a simple, meaningful blank-filling action (typing "Caucasus" on one of the entry lines – see Figure 43 and nearby discussion of Fig. 43.)

Those deeply nested functions in Fig. 3b – TextKAObjectFn, etc., may look a bit intimidating, but it's just boilerplate – a rule schema – that derives from Cyc having automatically generated a DCGS-MC CDD document section, a subsection of the section, a knowledge acquisition object (a field) for the subsection, and, finally, an object that results from a certain string being asserted by the user into the knowledge acquisition field in question.

In other words, the expression (TextKAObjectFn (CDDSubSectionOfSubSectionTypeFn (CDDSectionOfSectionTypeFn TestCDDDocument02 CDDIntroduction-PCW) CDDIntroductionSubSection-TitlePage-PCW) (KATemplateFieldOfTextTypeFn CDDIntroductionSubSection-TitlePage-PCW DODAgentAndInstrumentSystem 1) "Marine Corps Distributed Common Ground Surface System") is just a complicated-looking way to *denote* the Marine Corps Distributed Common Ground Surface System, which it learned about because the user entered a certain string ("Marine Corps Distributed Common Ground Surface System") on a certain field associated with a the Title Page subsection of the Introduction section of the CDD currently in development. If we replace this with an atomic expression for the sake of readability – MarineCorpsDistributedCommonGroundSurfaceSystem, or even better DCGS-MC, say, the intent behind the assertion may be more intuitive:

```
(thereExists ?AGENT
  (and
    (hasAgents  DCGS-MC  ?AGENT)
    (intendedTheaterOfOperations  ?AGENT  (TerritoryFnCaucasus))))
```

I.e., DCGS-MC has some agent(s) who can operate anywhere in the Caucasus.

As a result of SIDECAR asserting this to Cyc – entering it into Cyc's knowledge base – Cyc thereafter believes that fact to be true, at least until/unless it is later retracted or contradicted by other facts.

Now suppose the Cyc KB also contains the information that this system, DCGS-MC, is an instance of the class IntelligenceAnalysisAndProductionSystem. How does the Cyc Inference Engine conclude that DCGS-MC has an agent which can operate in Armenia?

Here is where the details of the definition of intendedTheaterOfOperations become very important. Among the other things that are asserted about this predicate, in Cyc, is the fact that it is *conservative* with respect to geographicalSubRegions via its second argument:

(conservativeViaArg intendedTheaterOfOperations geographicalSubRegions 2)

Figure 4. *A rule-abbreviating assertion as it appears in the Cyc KB Browser, using the rule abbreviating predicate* conservativeViaArg. *What this assertion means is that if an agent is intended to operate US-wide, such as the FBI, then (at least by default) it can operate in any part of the US such as Texas.*

The predicate conservativeViaArg is one of Cyc's rule-abbreviating predicates. The most important rule about it – in effect its definition or expansion – is this one (call it R1):

```
(implies
  (and
    (conservativeViaArg ?PRED ?BIN-PRED ?N)
    (trueSentence ?SENTENCE)
    (argN ?PRED ?SENTENCE  0)
    (argN ?ARGN ?SENTENCE ?N)
    (?BIN-PRED ?ARGN ?NEW-ARGN)
    (evaluate ?NEW-SENTENCE
      (SubstituteFormulaArgFn ?N ?NEW-ARGN ?SENTENCE))
  (trueSentence ?NEW-SENTENCE))
```

To put this into English: If (conservativeViaArg PRED BIN-PRED N) is true, and SENTENCE is any true sentence where PRED is in the arg0 (relation) position, and ARGN is the N^{th} argument, and it is also true that (BIN-PRED ARGN NEW-ARGN), then the new sentence that results from substituting NEW-ARGN for ARGN in SENTENCE is also true (hence inferable).

This sounds complicated, so let's see how it applies in the current case where PRED = intendedTheaterOfOperations and BIN-PRED = geographicalSubRegions and N=2. What R1 says in this case is: if (intendedTheaterOfOperations UNIT Caucasus) is true, and the Cyc Inference Engine can prove by any means, by any valid line of reasoning, that (geographicalSubRegions Caucasus SUBREGION) is also true, then together that forms a valid argument that (intendedTheaterOfOperations UNIT SUBREGION) holds

true[13]. I.e., if a unit has the entire Caucasus as its intended or targeted theater of operations, then it is intended to be operational in (or deployable to) any particular subregion of the Caucasus.

In fact, for many years Cyc has known that:

```
(properGeographicalSubRegions Caucasus-Region
  (TerritoryFn Armenia))
```

Figure 5. *A 'data level' assertion which has been in the Knowledge Base for many years, asserting the fact that the Caucasus region strictly subsumes the entire territory of Armenia.*

Cyc has also known for many years that the predicate properGeographicalSubRegions is a logical specialization of geographicalSubRegions.

Using those assertions, the Cyc Inference Engine is able to prove that the unit which is the agent of what we are calling the DCGS-MC and which has the Caucasus as its intended theater of operations, that that unit also therefore has the (TerritoryFn Armenia) – the geopolitical territory of Armenia – as a theater of operations.

The absolutely critical thing to see is that, because of the declarative use-neutral way the various pieces of supporting knowledge are represented in the Cyc KB, an analogous result can be proved for, e.g., the city of Baku (which is in Azerbaijan which is in the Caucasus). Cyc's geographical database is fairly comprehensive, having mapped NGA data bases to Cyc via a technique described in [Reed and Lenat 2002], and geographicalSubRegions is asserted in Cyc's KB to be a transitive binary predicate[14], so the result can be proven for a great many geographical areas, starting from that one initial fact about the intended theater of operations of DCGS-MC including the Caucasus.

---

[13] One might quibble about whether 'intended' as it appears in the *name* of this predicate is in fact captures the sense of the semantics: someone might *intend* a unit to be deployable to a certain region without being aware of all of the subregions. This is an example of why it's important to look at the content of the rules, not the *names* of the terms/rules. The semantics of the CycL terms are given by the assertions/rules involving them, *not* by the name of the term. We could have called that predicate g028153hello instead of intendedTheaterOfOperations and the conclusions would still follow – it would just be more difficult for a developer to find that term, later, if we gave it a highly non-mnemonic name like that.

[14] If R is transitive, and R(x,y) and R(y, z), then one can conclude R(x,z). So, for any regions x, y, and z, if (geographicalSubRegions x y) and (geographicalSubRegions y z), then (geographicalSubRegions x z).

3 The Underlying Semantic Technology (CYC Ontology)

There are three classes of CycL vocabulary associated with SIDECAR reasoning in connection with JCIDS documents:

1. vocabulary that supports representing and reasoning about JCIDS document substructures. E.g., the Title Page is a part of the Introduction section of a CDD.
2. vocabulary that supports recording user actions with respect to knowledge acquisition objects and inferring assertions on the basis of these actions in appropriate documentation contexts. E.g., the rule of thumb that says:
   If the Interest Designation of the CDD is a US military organization,
   Then the Validating Authority is likely to be that same military organization.
3. vocabulary that supports representing background knowledge regarding systems, system requirements, capabilities, and capability-related dependencies. E.g., if some component of the system SIDECAR is being told about must support communication among wireless networked devices, then here are the 4 latest set of approved IEEE 802.11 communication standards to choose from, along with requirements, costs, and other parameters of each.

We'll briefly describe each of these vocabulary sets in turn in Sections 3.1 – 3.3:

3.1 Document Representation

JCIDS documents typically exhibit highly regular outline structures, based upon their type. For example, a Capability Development Document (CDD) has a mandatory structure consisting in an introductory section plus 16 numbered sections and three required appendices.

Most sections have additionally mandated substructure: e.g., the CDD introduction is supposed to consist in a title page, executive summary, table of contents, and points-of-contact section.

This regularity makes it feasible for us to auto-generate a CDD[15] using CycL rules and specialized functions for generating the terms that represent the subsections of a particular document. Rules and specialized functions are also used for generating the *dynamic SIDECAR interaction context* where all of the user's interactions with elements of the document-specific knowledge acquisition scheme are recorded (e.g., what choices they made, and in what order), and for generating the multiple inheritance hierarchy of reasoning contexts associated with the various elements of the document. See Figure 6, below.

[15] Actually what this produces is the full internal formal representation of the content of each section and subsection of the CDD. Farther on we discuss how SIDECAR automatically converts this model -- these CycL assertions -- into a human-readable textual English CDD.

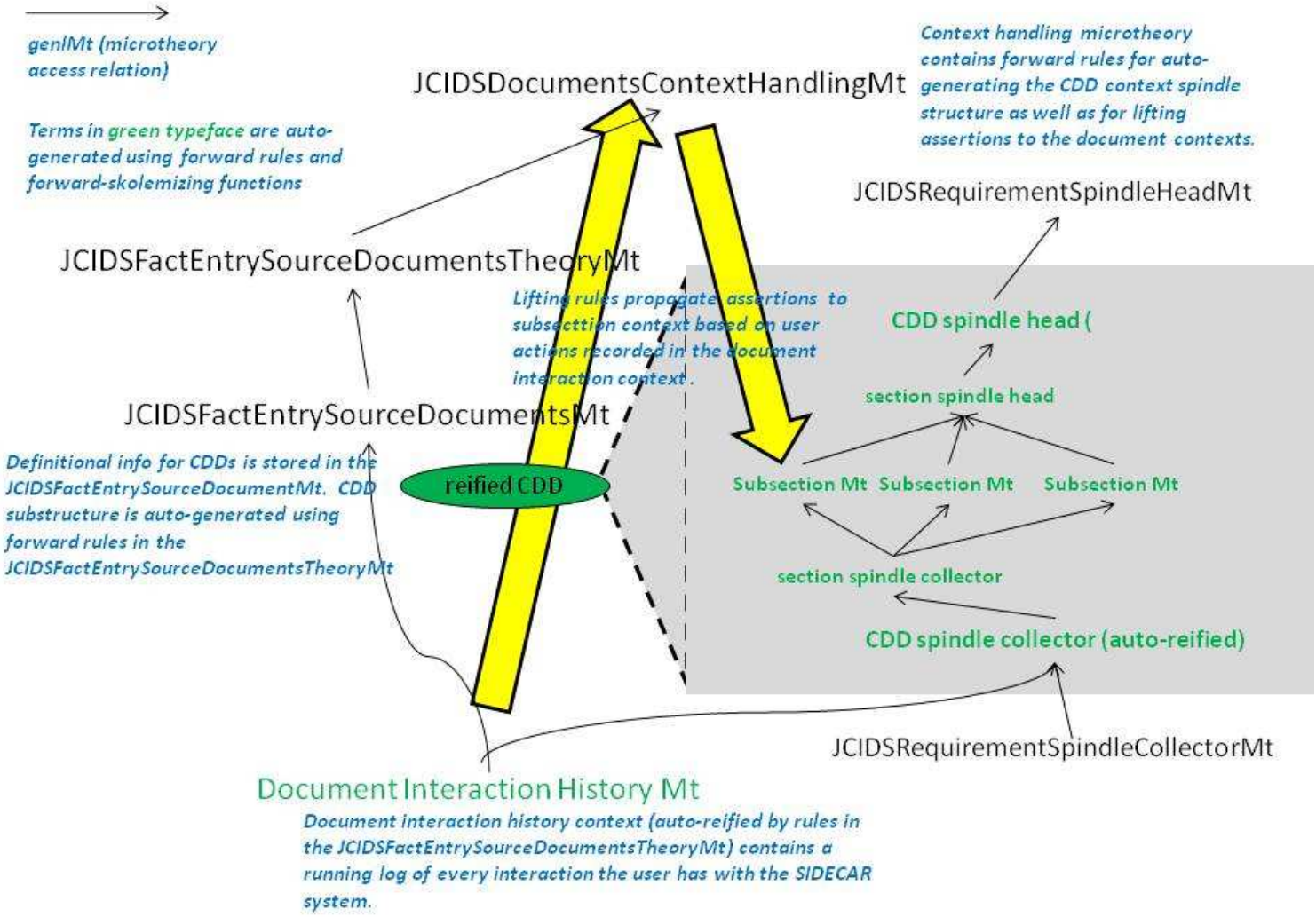


Figure 6. *A schematic showing the generation of the document substructure for a hypothetical CDD. Rules for auto-generating the document substructure are stored in the JCIDSDocumentContext-HandlingMt, while rules for interpreting the knowledge acquisition objects of the CDD knowledge acquisition scheme are stored at present in the JCIDSFactEntrySourceDocumentsTheoryMt. The document itself, together with its auto-generated substructure, is stored in the JCIDSFactEntrySource-DocumentsMt. Associated with each 'leaf node' of the document substructure is a document reasoning context: these are fitted into a so-called microtheoretic 'spindle structure' which has the JCIDSRequirementSpindle-HeadMt and JCIDSRequirementSpindleCollectorMt as its permanent access points. The interaction history microtheory that SIDECAR automatically generates for the document 'sees' (and treats as known and true) the content of both the contextual spindle structure and the aforementioned JCIDSFactEntrySourceDocumentsTheoryMt, whose rules specify where in the spindle assertions concluded from user interactions with the knowledge acquisition scheme are to be lifted. Terms which appear in black in the diagram are permanent parts of the Cyc KB, while terms appearing in green are auto-generated at run time. The 'Mt' suffix appearing on many of these CycL terms is short for 'microtheory' and denotes a reasoning context.*

At present, SIDECAR can produce a full instance of this formal model -- this substructure and set of reasoning contexts -- only for Capability Development Documents. But our methodology should directly extend to any of the other JCIDS document types such as ICDs. Here, in Figure 7, we see what the top-level document representation looks like in the Cyc KB for a particular CDD document, namely the CDD describing DCGS-MC:

Figure 7. *Top-level CDD sections of the DCGS-MC represented in SIDECAR. The Cyc Browser is a GUI that makes the display of assertions very compact; here there are dozens of separate assertions represented using a small amount of "screen real estate".*

Figure 8. *Some of the textual components of subsections of the DCGS-MC CDD represented formally in Cyc's KB by SIDECAR (as compactly displayed here in the Cyc KB Browser).*

The sixteen CDD sections are visible in Figure 7, and Figure 8 delves deeper to show a fraction of the subsection-level textual components. The binary relation cDDTextualComponent connects the document to its sections, and also connects a section to its subsections. Although the information is not shown in this view, the KB also includes an explicit specification of the section ordering and also an explicit specification of the orderings of the sub-sections within the sections. These orderings help the English document generation subsystem of SIDECAR to arrange the automatically generated English sentences into a natural order, and where to place section heading, subsection headings, etc., so that the final result is a familiar-looking, formatted, readable final CDD in English.

The green 'bullets' or spheres that one sees in Figures 7 and 8 are a user interface icon – a convention used by the Cyc KB Browser – to specify that a particular result was remotely concluded – that is, derived by means of a forward rule. We rely heavily on forward rules in the Knowledge Base for the purpose of generating the substructure for the document under development. It will be noted that we also rely heavily on functions and their resultant NARTs, to represent the CDD components as first order objects[16].

3.2 Knowledge Acquisition Framework

The knowledge acquisition framework created for SIDECAR starts with the fundamental insight that the task of correctly interpreting information being dynamically entered via a structured 'template' shares many important characteristics with the task of correctly interpreting structured data preserved in a schematized database.

This problem could be solved through some NxN data warehouse-like mapping approach, but that quickly gets too costly to scale up; instead, we map each schema element of each database to Cyc, using the Cyc ontology as a sort of interlingua, which converts the problem from quadratic cost N^2 to a linear cost N. This technique is what we refer to as Cyc Semantic Knowledge Source Integration (SKSI); please see, for a detailed description of this technology in theory [Reed and Lenat 2002] and in practice [Masters 2002].

Character strings entered or selected in association with knowledge acquisition objects (fields, menus, interactive diagrams, etc.) are analogous to string values entered in the physical fields of a database. In the same way that that Cyc meta-vocabulary is deployed in SKSI to identify logical field values and interpret them according to semantic templates associating logical field indexicals, meta-vocabulary is used by SIDECAR in the course of JCIDS-related knowledge acquisition to create semantic templates associating knowledge object indexicals, which in turn are used in interpreting actual string entries or selections that are made in association with said objects. The salient distinctions between "traditional" Cyc SKSI the current SIDECAR mappings are:

[16] 'First order object' is something of a term of art among ontologists: it simply means a term that can be bound to an open variable in a query, or indirectly qualified by a quantified expression. The analogue in English is: which concepts are namable by one English word, versus requiring a phrase.

- Dynamic Knowledge Acquisition requires identifying (and specifying the behavior of) many more different kinds of knowledge acquisition objects other than fields: notably (but not limited to) menus, dialog boxes, and interactive diagrams; and
- Because there are relatively fewer constraints on how strings are used to represent terms as compared with a database, a much higher premium is placed on front-end natural language understanding of the user's input.

3.2.1 Knowledge Acquisition Objects and Inference in Cyc

The SIDECAR Knowledge Acquisition vocabulary has been defined with the intent of affording users flexibility in defining knowledge capturing applications. For a given class of document (e.g., the Capability Development Document), Cycorp ontologists have considerable leeway for defining the knowledge acquisition scheme that will be expressed in the user interface (UI), a process that is undertaken based on consultation with subject-matter experts, study of example documents, and review of the JCIDS operations manual[17]. As noted above, four different kinds of knowledge acquisition object (henceforth, KA object) have been defined: fields, menus, dialog boxes, and interactive diagrams. The concepts of the various types of *action* that a user can take with respect to a given KA object (entering a string value on a field, say), are defined in the Knowledge Base as well, together with various salient attribute 'slots': what string was actually entered, *when* it was entered, who entered it, what field the value was entered on, what text section is the source for the user's assertion, and so on. Whenever the user takes a permitted action with respect to a KA object, this action is automatically represented in the aforementioned interaction context that the system generates for the document under development, with all of the values for the attribute slots supplied. This enables SIDECAR to maintain a complete chronology of all of the permitted information-supplying actions that a user takes in entering knowledge from a document: not only *entry* of knowledge (adding a field value, selecting an option from a menu), but also *corrections* of prior entries (deleting an entry in a field, deselecting a previously selected menu item), so that at any time, Cyc has available both a complete record of a document's development in SIDECAR and a representation of the current states of all KA objects used in collecting information about the document to reason from. We could even record the user's every mouse movement and the speed with which they type every character – and have done so for some intelligence analyst modeling applications in the past – but do not go down to that level for SIDECAR currently.

In addition to this infrastructure, we specify what are called 'meaning sentences' that tell the Cyc system how to interpret entries on various combinations of KA objects. For example, in the 'Points-of-Contact' subsection of the Introduction section of a Capability Development Document, we include fields for entering the name of the capability officer as well as the name of the organization the capability officer represents.

---

[17] United States Department of Defense. *Manual for the Operation of the Joint Capabilities Integration and Development System*. 31 July 2009.

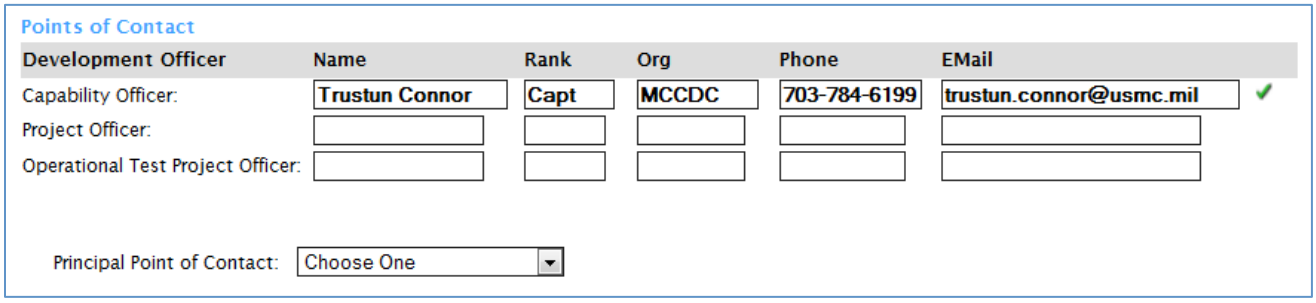


Figure 9. *The principal point of contact section from the tabbed view of the CDD Introduction Section in the SIDECAR interface.*

In addition to these fields, however, an assertion is included in the document model that says that the person named in the capability officer name field is – presumably! – a member of the organization named in the capability officer organization field. We can then write a general rule leveraging from this (and similar interpretation sentences in the Knowledge Base) to lift an appropriate assertion into the appropriate documentation context, if and when a person and an organization are recognized via the string infills on the aforementioned fields. Because the KB-state-reporting relations used to write these rules update dynamically based on the state of the interaction context, the lifted assertions in the documentation contexts are also updated based on actions that the user takes with respect to the relevant KA objects. The KA objects and associated truth maintenance rules are defined in such a way as to afford Cycorp ontologists considerable say over the circumstances in which an update will happen. For example, fields are defined so that if a new string-entering event occurs in relation to a field where there has already been a string entering event, the old string entry will be overwritten. To take an even more extreme example, if – via some Marine Corps personnel data base, for example – the status of one of these points of contact changes, and they are no longer with their prior organization, e.g., then Cyc's truth maintenance capability would detect that new contradiction, and SIDECAR's policies might cause it to "flag" this CDD for revisiting and updating.

Some of the most constrained choices in SIDECAR are crystallized into menus, but even there, as things change "in the world", the assertions which led SIDECAR to populate a menu with various choices will cause it to produce a corrected updated set of alternative choices automatically. Menus are defined so that a previous user selection must be explicitly deselected in order to 'undo' assertions generated by a previous selection from the menu. Some menus can and should allow only a single selection; other menus might allow two or any number of selections, and so on. Again, this is something that SIDECAR can reason about, and changes (in policy, technology, etc.) can cause those constraints to automatically produce updated, corrected behavior of the runtime SIDECAR system.

An example follows, and is discussed as we present Figures 10-18, below. To lay the groundwork for that example, consider that the working SIDECAR prototype for CDD creation includes a section for entering the principal points of contact (POCs); for each one, the specification says that the user might enter (i.e., the model includes fields for)

that person's name, rank, organization, phone number, email address, etc., for each POC: for the capabilities officer, the project officer, and the operational test project officer.

Now suppose a user enters a name, "Scott Camden", and a rank ,"LtCol.", in the respective name and rank fields for the Project Officer. Both of these data-entry events are automatically recorded in the KB. The respective records for the name entry and the rank entry are shown below in Figures 10 and 11. This includes who entered the data, when exactly they entered it, etc.

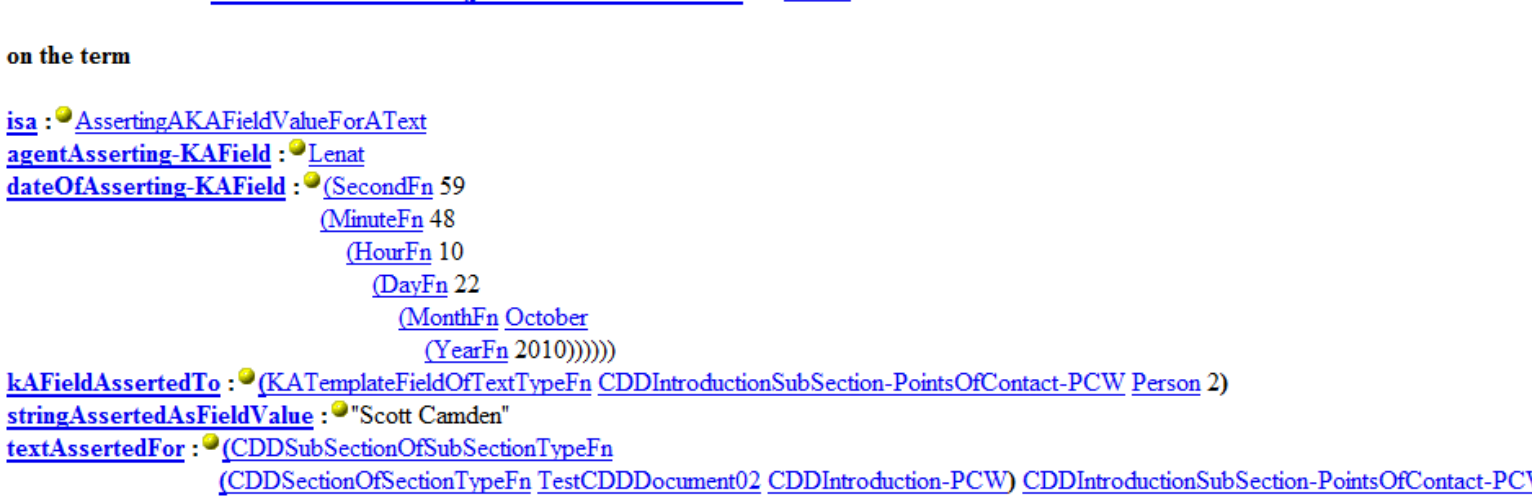


Figure 10. *Cyc KB representation of the act of entering a name string in the project officer 'name' field of points-of-contact subsection of the DCGS-MC CDD.*

Figure 11. *Cyc KB representation of the act of entering a rank-denoting string in the project officer 'rank' field of points-of-contact subsection of the DCGS-MC CDD.*

Both of these are instances of a collection called 'AssertingAKAFieldValueForAText', a specialization of Cyc's concepts of PurposefulAction. Notice that we record the time of entry, the author, the text subsection for which the assertion is made, and the text string that is entered.

A forward rule in the KB then generates KB state-reporting assertions for each of these entries, directly linking the field, the text subsection, and the string that is entered:

```
(kAFieldHasStringValueInText
   (KATemplateFieldOfTextTypeFn CDDIntroductionSubSection-PointsOfContact-PCW MilitaryPersonTypeByRank 2)
   (CDDSubSectionOfSubSectionTypeFn
      (CDDSectionOfSectionTypeFn TestCDDDocument02 CDDIntroduction-PCW) CDDIntroductionSubSection-PointsOfContact-PCW) "LtCol")
in (JCIDSDocumentInteractionHistoryMtFn TestCDDDocument02)
(kAFieldHasStringValueInText
   (KATemplateFieldOfTextTypeFn CDDIntroductionSubSection-PointsOfContact-PCW Person 2)
   (CDDSubSectionOfSubSectionTypeFn
      (CDDSectionOfSectionTypeFn TestCDDDocument02 CDDIntroduction-PCW) CDDIntroductionSubSection-PointsOfContact-PCW) "Scott Camden"
in (JCIDSDocumentInteractionHistoryMtFn TestCDDDocument02)
```

Figure 12. *Remotely concluded KB state-reporting assertions for the project officer name and rank fields relative to the DCGS-MC CDD points of contact subsection.*

The rule that supports these state reporting assertions is written with an exception that overrides its result if and when an AssertingAKAFieldValueForAText instance is known such that the field and text subsection (the kAFieldAssertedTo and the textAssertedFor) are the same, and the timestamp of the entry (dateOfAsserting-KAField) is *later* than the timestamp of the entry that would otherwise trigger the rule. A separate forward rule requires Cyc to reconsider deductions on all assertions involving kAFieldHasStringValueInText whenever a new AssertingAKAFieldValueForAText value is entered on a given field and text section. The combined effect is to allow the user to overwrite the content of a field, with the KB state report updating accordingly.

Meanwhile, the KB also includes an assertion that specifies how to interpret the string entries on these fields, as part of the 'document theory' of CDDs:

```
(meaningSentenceOfTextType CDDIntroductionSubSection-PointsOfContact-PCW
  (rank-Military
    (TheKAFieldValueMappedFn
      (KATemplateFieldOfTextTypeFn CDDIntroductionSubSection-PointsOfContact-PCW Person 2))
    (TheKAFieldValueMappedFn
      (KATemplateFieldOfTextTypeFn CDDIntroductionSubSection-PointsOfContact-PCW MilitaryPersonTypeByRank 2)))
```

Figure 13. *KB assertion specifying the relation between the objects denoted by the entries in the project officer name and rank fields.*

The semantics of this are such that it says, in effect, that the thing (the person) which the entry in the project officer name field refers to has the military rank (rank-Military) that is referred to by the entry in the project officer rank field. Cyc then uses a forward rule (shown in Figure 14):

EL Formula :

Mt : JCIDSFactEntrySourceDocumentTheoryMt

```
M(implies
    (isa ?PCW-TYPE ContextGeneratingJCIDSDocumentSubSectionType)
    (trueRule
      (CollectionRuleTemplateFn ContextGeneratingJCIDSDocumentSubSectionType)
      (implies
        (and
          (isa ?PCW ?PCW-TYPE)
          (properNameStrings ?THING-1 ?STRING-1)
          (properNameStrings ?THING-2 ?STRING-2)
          (contextOfPCW ?PCW ?PCW-MT)
          (theKAFieldValueMappedIndexical ?FIELD-1 ?INDEX-1)
          (theKAFieldValueMappedIndexical ?FIELD-2 ?INDEX-2)
          (kAFieldHasStringValueInText ?FIELD-1 ?PCW ?STRING-1)
          (kAFieldHasStringValueInText ?FIELD-2 ?PCW ?STRING-2)
          (meaningSentenceOfTextType ?PCW-TYPE
            (?PRED ?INDEX-1 ?INDEX-2)))
        (ist ?PCW-MT
          (?PRED ?THING-1 ?THING-2)))))
```

Figure 14. *An example of a general, interpretive, assertion-lifting rule template in the Cyc KB.*

This rule says that for any type of JCIDS document sub-section, if two fields have respective string value entries for a text subsection that is an instance of that type, and the objects those fields refer to are linked by a binary predicate in a meaning sentence for the subsection type, and the Cyc system can identify referents (via properNameStrings) for the strings entered on the fields, then the sentence formed by substituting these referents into the meaning sentence for the text type (with the predicate in the arg0 position and the referents in the indicated arg1 and arg2 positions) should be asserted (lifted) into the documentation context associated with the text.

The consequence of this rule, plus the field entries and automatically inferred KB state-reporting assertions just described, is that SIDECAR concludes that Scott Camden has the rank of LieutenantColonel-Rank; that assertion gets entered into the Cyc KB, namely into the documentation context that the KB associates with the points-of-contact subsection of the introduction section of the DCGS-MC CDD.

The predicate properNameStrings in the forward rule is worth remarking on: this supports natural language (NL) reasoning from the string to the CycL term it represents.

For example, the Cyc KB has sufficient lexical knowledge to infer several character string denotations for the military rank represented in CycL by the term LieutenantColonel-Rank: e.g., 'LtC.', 'LtCol', 'Lieut. Col.', 'Lieutenant Colonel', etc. Entering any one of these strings in the rank field will generate a mapping to LieutenantColonel-Rank. It is also worth remarking that the name fields of the points of contact subsection are of a special character in this respect: if the Knowledge Base already knows about exactly one person with the name entered, results from the corresponding POC entries will be concluded for that person, but if no person is recognized, an instance with the name will be automatically generated. This 'auto-generating' feature is characteristic of many of the knowledge acquisition objects used in the SIDECAR prototype: namely, all objects for which there is reason to think the KB would not already include every term that a user might use the KA object to refer to. Among the other KA objects with this feature is the title page field that the user employs to enter the name of system being described.

It is important to emphasize that, once assertions have been 'lifted' into documentation contexts via interpretive inferences of the sort just described, this information becomes available to Cyc reasoning, just like any other information in the Cyc Knowledge Base. Thus, for example, once ranks have been entered for several development officers and a principal point of contact (PPOC) has been identified, we can ask a number of queries that rely on this information plus additional background knowledge, including queries that ask for information that is only *implicit* in the information that has been provided in the context of the body of background knowledge that is being drawn upon.

For example, we can query for documents, persons, and ranks where the principal point of contact for the document is outranked by some other development officer. Figure 15 shows the CycL version of that query, but we can use an English query-construction tool such as the one we developed for Cleveland Clinic doctors to pose *ad hoc* queries, so that SIDECAR users can get questions asked (and answered) in English without having to know anything about Cyc or formal logic [Lenat *et al* 2010].

```
(and
  (citesAsPrincipalPointOfContact ?DOC ?PPOC)
  (citesAsDevelopmentOfficer ?DOC ?SOMEONE-ELSE)
  (rank-Military ?PPOC ?LOWER-RANK)
  (rank-Military ?SOMEONE-ELSE ?HIGHER-RANK)
  (followingValueOnScale ?LOWER-RANK ?HIGHER-RANK USMarinesRank))
```

Figure 15. *A query formula that asks for documents, persons, and rank such that one person is a cited as a development officer by the document, and the other is the cited PPOC, where the rank of the PPOC is less than that of the other development officer in the USMC.*

Suppose that a capabilities officer and a project officer have been identified for the document under development , with the respective names of 'Trustun Connor' and 'Scott Camden', and the respective ranks of Captain and Lieutenant Colonel. Thanks to the

automatic assertion lifting such as has been described, the Cyc Knowledge Base has the information that Camden is a cited project officer for the document, and Connor is the cited PPOC:

```
(citesAsProjectOfficer TestCDDDocument02
  (TextKAObjectFn
    (CDDSubSectionOfSubSectionTypeFn
      (CDDSectionOfSectionTypeFn TestCDDDocument02 CDDIntroduction-PCW) CDDIntroductionSubSection-PointsOfContact-PC
    (KATemplateFieldOfTextTypeFn CDDIntroductionSubSection-PointsOfContact-PCW Person 2) "Scott Camden"))

(citesAsPrincipalPointOfContact TestCDDDocument02
  (TextKAObjectFn
    (CDDSubSectionOfSubSectionTypeFn
      (CDDSectionOfSectionTypeFn TestCDDDocument02 CDDIntroduction-PCW) CDDIntroductionSubSection-PointsOfContact-PCW)
    (KATemplateFieldOfTextTypeFn CDDIntroductionSubSection-PointsOfContact-PCW Person 1) "Trustun Connor"))
```

Figure 16. *KB representation of the fact that Scott Camden is the cited Project Officer for the DCGS-MC system described in this CDD, and Connor is the cited Principal POC.*

The predicate citesAsProjectOfficer is a specialization of citesAsDevelopmentOfficer, so that the Cyc Inference Engine is able to prove that Scott Camden is cited as a development officer by the document. In addition, background knowledge includes the fact that the rank of Captain follows the rank of Major on the USMC scale, and the fact that the rank of Major follows the rank of Lieutenant Colonel.

```
(followingValueOnScale Captain-Rank Major-Rank USMarinesRank)
(followingValueOnScale Major-Rank LieutenantColonel-Rank USMarinesRank)
```

Figure 17. *Background knowledge long-known to Cyc about US Marine Corps ranks.*

The KB also includes a rule that asserts the transitivity of the followingValueOnScale relation with respect to its first two arguments: if Y follows X on a given scale, and Z follows Y on the same scale, then Z follows X on the scale:

```
'(implies
   (and
     (followingValueOnScale ?ATT1 ?ATT2 ?SCALE)
     (followingValueOnScale ?ATT2 ?ATT3 ?SCALE))
   (followingValueOnScale ?ATT1 ?ATT3 ?SCALE))
```

Figure 18. *Rule asserting transitivity on the first two arguments of followingValueOnScale.*

This enables the Inference Engine to prove that Scott Camden's rank of LieutenantColonel-Rank exceeds Trustan Connor's rank of Captain on the USMC scale.

Thus, the document in development, the four open variables of our query are satisfied by the quadruple of values **<Trustun Connor, Scott Camden, LtCol, Captain>**.

This illustrated how information builds on information in the Cyc Knowledge base: a plethora of rules and rule-abbreviating relationships allow the user to conclude a significant wealth of data even with respect to concepts and individuals whose denoting terms are 'newly minted', provided these can be appropriately situated in the Cyc ontology.

For example, in addition to tying the development officers to the document in which they are cited, we also link them to the system they are development officers of, in their respective capacities. From this, additional inferences can be drawn provided there is adequate background knowledge about the Joint Force system development – e.g., we might infer what roles the cited individuals might play in various events that could be expected to occur in the history of the proposed system, such as milestone reviews.

Cyc reasons by *argumentation*, not just propagating statistics (via Bayes' Rule) nor propagating True and False tokens (via Frege). It gathers up all the pro and con arguments it can muster, for any hypotheses, and then uses tactical and strategic heuristics to decide which arguments "trump" which others. For example, here are arguments it assembled shortly after the 2004 Madrid train station bombing, about whether ETA (versus Al Qaida or some third group) was most likely responsible for it:

For:

- ETA often executes attacks near national election
- ETA has performed multi-target coordinated attacks
- Over the past 30 years, ETA performed 75% of all terrorist attacks in Spain
- Over the past 30 years, 98% of all terrorist attacks in Spain were performed by Spain-based groups, and ETA is a Spain-based group.

Against:

-ETA warns (a few minutes ahead of time) of attacks that would result in a high number civilian casualties, to prevent them. There was no such warning prior to this attack.
-ETA generally takes responsibility for its attacks, and it did not do so this time.
-ETA has never been known to falsely deny responsibility for an attack, and it did deny responsibility for this attack.

Figure 18b. Cyc's pro and con arguments for ETA being responsible for the 2004 Madrid train station bombing. Different individuals and groups had (a) different states of knowledge/informedness, often changing hour by hour, and (b) different sets of heuristic rules based on their ideology, nationality, age, etc;, and these differences all contributed to the lack of consensus, and downright confusion, at the time.

3.2.2 Implications of the JCIDS Operational Context

The analytic basis and core of JCIDS, and the basis for evaluating and validating requirements and results in the potential development and deployment of new or improved capabilities is the Capabilities Based Assessment (CBA) process as detailed in the *Manual for the Operation of the Joint Capabilities Integration and Development System* [United States Department of Defense, July 2009, p. 7-8]. The intended operational outputs of a CBA are:

- Description of the mission and military problem being assessed.
- Identification of the tasks that need to be completed in order to meet the mission objectives.
- Identification of the required operational capabilities.
- Assessment of how well the current or programmed force meets the capability needs.
- Assessment of operational risks where capability gaps exist.
- Recommendations for non-material solutions to identified capability gaps.
- Recommendations for potential material approaches to identified capability gaps

CBAs unfold within a large context of strategic and policy guidance documents centered on the National Security Strategy, the National Strategy for Homeland Defense, the National Defense Strategy, and the National Military Strategy and including (but by no means limited to) the Guidance for the Development of the Force, the Guidance for the Employment of the Force, and the latest Quadrennial Defense Review Report, [United States Marine Corps, July 2009, p. 8], and in turn require generation of a series of increasingly-more-specific documents characterizing the capabilities of the proposed system and their relation to known Joint Force requirements: notably an Initial Capabilities Document (ICD), a Capability Development Document (CDD) and a Capability Production Document (CPD). The expectation is that the document content will reflect and accommodate criticisms and observations introduced by the ongoing review process, in addition to addressing relevant Joint Force policies, guidance, and known operational requirements as detailed by the aforementioned operational context of strategic policy and guidance documents.

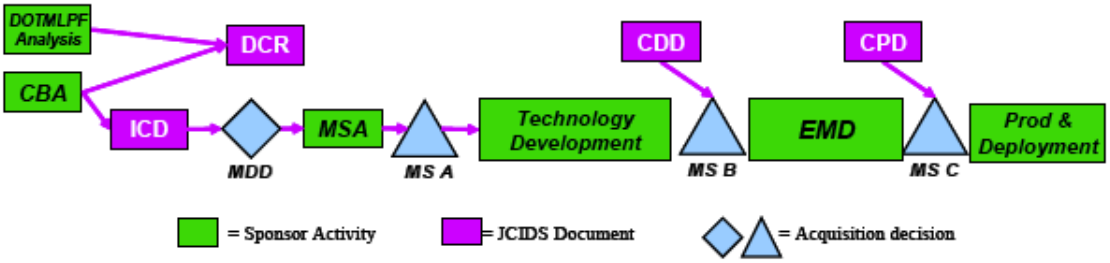


Figure 19. *The ideal Capabilities Based Assessment workflow.*

This framework gives rise to a strict requirement that the documentation generated (ICD, CDD, etc.), in order to be effective, must reference relevant policy, theory, and doctrine in explicating the architecture of the proposed system: that is, the architecture can't merely be *described* as a *fait accompli*: it must be *justified* in light of relevant requirements and doctrine. In addition, the CBA framework creates a second, no-less-important premium on the early discovery of situations wherein documents developed in the course of parallel CBAs turn out to have implications for each other. This includes not only the obvious case, where documents make conflicting claims, but also subtler cases wherein the systems proposed by the respective documents are a good operational 'fit' for one another, or where a proposed system might even be subsumed as an operational component within another existing or proposed system.

The difficulty, of course, is that the theoretical framework underlying Capabilities Based Assessment already encompasses more documentation and rules than a single human author of an ICD, CDD, or CPD can reasonably be expected to assimilate within the time frame allowed for the creation of the document (never mind the penumbra of potentially *related* documents that would need to be perused in order to establish real coverage on the compatibility/conflict front). The SIDECAR prototype addresses these problems directly, by incorporating within the Cyc Knowledge Base Server a formal version of the theoretical framework stipulating current doctrine concerning the satisfaction of functional system requirements, in addition to making provision for storing formal models of the contents of various documents generated by the CBA process, that we acquire through SIDECAR-based Knowledge Acquisition. These models can be used, along with Cyc inference and authored, KB-side knowledge acquisition schemas, to drive 'mixed initiative' knowledge acquisition interactions which not only capture descriptions of the proposed system architecture in an ICD, CDD, CPD, etc., but do this in a way that allows SIDECAR to tie capability implementations to requirements within a framework of known doctrine, strategy, and theory.

The Capability Development Document affords many potential examples. The CDD is an entrance criteria item needed to proceed to a milestone acquisition decision, and is required to describe the operational capability of the proposed system together with the countered threat, roles within the DOD Enterprise Architecture and relevant solution architectures, required capabilities, program support, sustainment, force structure, policy impacts, and constraints [United States Department of Defense, July 2009, p. 86]. Using the document section model, relevant policy documents such as the MAGTF Gap List [United States Navy, May 2007], and example CDDs, the knowledge acquisition schemas for the various CDD sections can be authored with engineered dependencies between appropriate Knowledge Acquisition objects with multiple entry points whereat the Cyc KB Server may guess about the position of the proposed system with respect to the DOD Enterprise Architecture; having made and confirmed such a guess, it can then populate additional, relevant KA objects (particularly diagrams and menus) in succession in order to help guide the user in identifying appropriate capabilities to cite, appropriate requirements engendered by these capabilities, and doctrinally sanctioned means of satisfying these requirements. Several examples of how such an engineered Knowledge Acquisition Scheme works in the prototype example may be found below, for example,

in sections 4.7, 4.10, and 4.11, where, having first confirmed a guess at a subsuming system based on the user's principal point of contact entries, SIDECAR uses this confirmation to help identify candidate architectural roles for the system. Having identified one such role, SIDECAR then uses this information in conjunction with known policy to identify several requirements for the system, candidate infrastructure capabilities for implementing these requirements, and, finally candidate device types for satisfying the capabilities.

3.3 Functionality Model

Behind much of the Knowledge Acquisition infrastructure is a vocabulary for representing and reasoning about system capabilities and functionality which enables both representation of and reasoning about many of the conclusions that need to be drawn from user interactions with KA objects. *Operational systems,* such as are described in JCIDS documents (e.g., the DCGS-MC, the MCISR-E, the P-ISR, etc.) are represented as complex, compositional, functional assemblages featuring both organizational and artifactual components, each of which may in turn be decomposed into subcomponents. For example, the DCGS-MC is associated both with an organization, composed of assigned and attached USMC units, analysts, etc., who support the operation of the system in various capacities, and also with a system of artifacts: a computer network (at least one), databases, personal communication devices, and perhaps sensors of various kinds. Systems are also associated with the processes implemented by various of their components, with the totality of these processes comprising the functional lifetime of the system.

Systems, along with their components, may feature both *requirements* and *capabilities*. Capabilities as a rule are modeled as 3-term relations between an agent, an event type, and a role, the idea being that, under 'normal conditions' (a concept which often remains under-specified), the agent is able to play the role in instances of the event type. A requirement can also be modeled as an analogous 3-term relation. The relationship between capabilities and requirements is modeled provisionally as follows: an agent's having a capability, in the aforesaid sense of being able to play a role in a certain class of functional situations, may entail certain requirements for the agent, in the sense that the agent will need components which are capable of playing situational roles in their own right with respect to the functional operation of the system. Capability-requirement dependencies are partly a function of physical and organizational principles, but also partly a matter of the stipulative policy conditions of the operational environment: the Joint Forces or other salient authorities may simply *mandate* that a particular capability be implemented through the satisfaction of certain requirements – for example, that a capability for disseminating reports, or a certain *kind* of reports, requires use of a computer network, or computer networks.[18] Starting from basic knowledge of capability

[18] A good example of a document that stipulates such dependencies is the 'MAGTF Gap List' (United States Navy. *Approved Solution Planning Directive (SPD) for Program Objective Memorandum (POM) - 10 Warfighting Investment Program Evaluation Board*. 3 May 2007,) Not only are requirements for

requirement dependencies, we can infer various kinds of requirements for a system based on knowledge of its situational roles within its larger concept-of-operations: given those requirements in the form or role/situation type pairings, the Cyc system can ask what requirements *they* entail as presumptive capabilities of the system, and so on. At any point in the analysis, Cyc can also check what kinds of things might have the capability that has been identified – thus enabling SIDECAR to serve as the driver for a controlled capability-requirement analysis starting from identifiable principles, with potential for use in system architecting and design. Examples can be found below, in the section describing our working prototype.

The Cyc ontology already includes a number of other important relationships that figure in the recursive functional organizations of Joint Force systems: for example, partonomic relations that hold between systems and their components (including *organizational* roles in contrast with situational ones), sub-situation relations that hold between system processes and their subprocesses, and inter-agent relations that hold between components in virtue of their situational and organizational roles within the system.

MAGTFs and supporting system classes identified, but direction is given as to what is viewed as a satisfactory infrastructure implementation for the requirement.

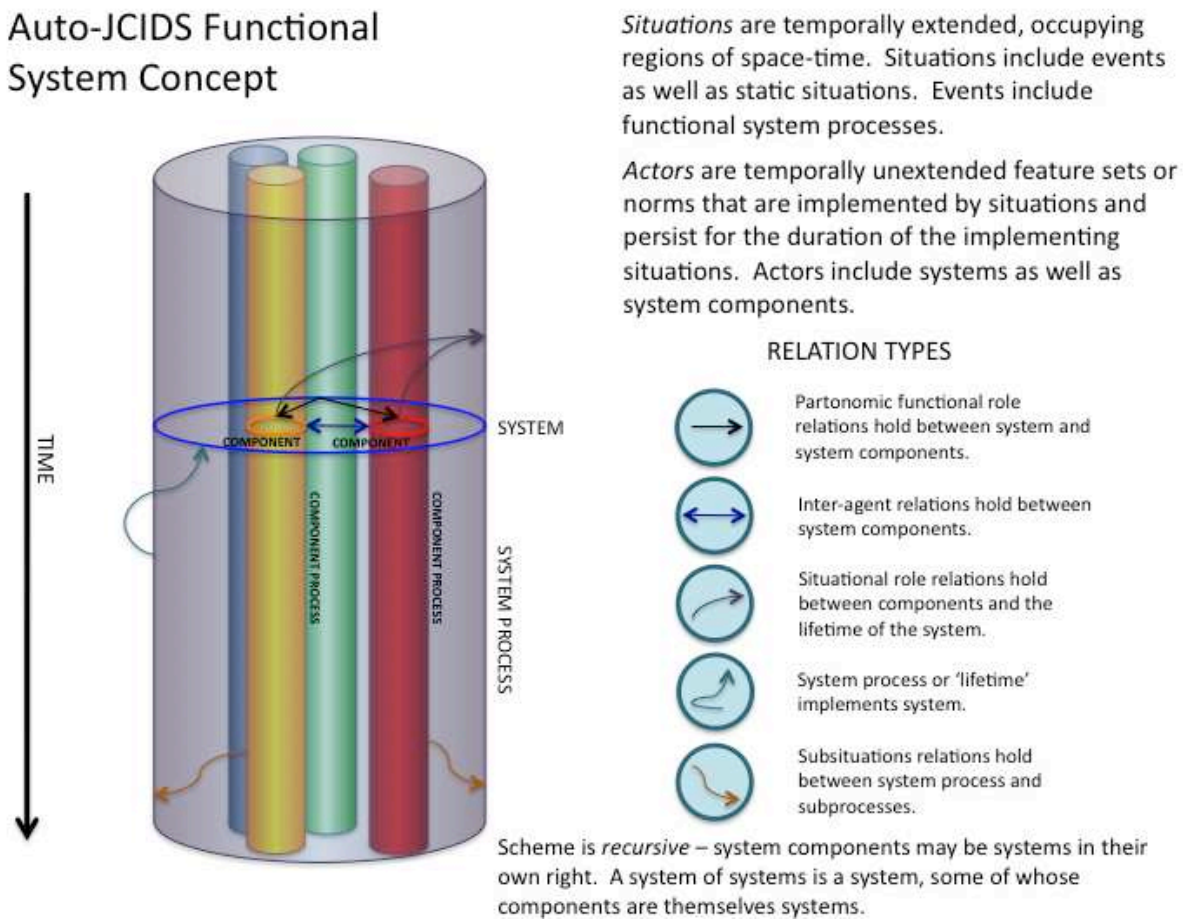


Figure 20. *The functional systems concept for SIDECAR, together with a listing of the relationship types covered in the model.*

4 Extensible Working Model Use Case

One way to explain the potential of the SIDECAR prototype is to walk through a demo of it running, discussing various salient aspects of what it's doing, and how, for a particular use case.

The illustrations on the pages that follow are live screen captures from such a case – namely, a use of the SIDECAR system to generate portions of a CDD for the Marine Corps Distributed Common Ground-Surface System (DCGS-MC). These examples cover portions of the Introduction, Section 1, Section 3, and Section 6 of the CDD, and the Glossary.

In the course of this walk-through, the reader will see explicit examples of the key SIDECAR AI functionalities:

- pro-active guidance of the user: suggesting what question to focus on next, what section to focus on next, etc.
- automatic constraining of choices they need to make
    - One important case of this is where the number of options drops to *zero;* in that case, SIDECAR completely removes a query/menu
    - Another important case is where the number of options drops to *one*, so SIDECAR can just fill in that choice as the answer for that query/menu
- detecting contradictions of various degrees of severity, and offering to help the user remedy those problems (or, if they choose, defer attending to it; sometimes, later-entered information may help – or completely – resolve the contradiction.)

4.1 Home Screen

The user is first taken to a splash screen displaying relevant logo's (currently, for demonstration purposes, these include the Cycorp logo together with logos for the United States Marine Corps and US Marine Intelligence):

This functions as the SIDECAR home screen. Available functions include a 'tabbed' view, a guided view, and a document view, in addition to a settings section.

The 'Section Classification' field, at the top of each CDD section's SIDECAR screen, shows the highest document classification referenced in that current section tab. The 'CDD Classification' field shows the highest document classification for any section tab completed thus far. For manual entries, the user should explicitly fill in a compartmentalization for any CDD classification higher than 'unclassified'. For automatic SIDECAR-inferred entries, classification level and compartmentalization may be inferred automatically by SIDECAR from the classification of the data sources and elements used in that inference.

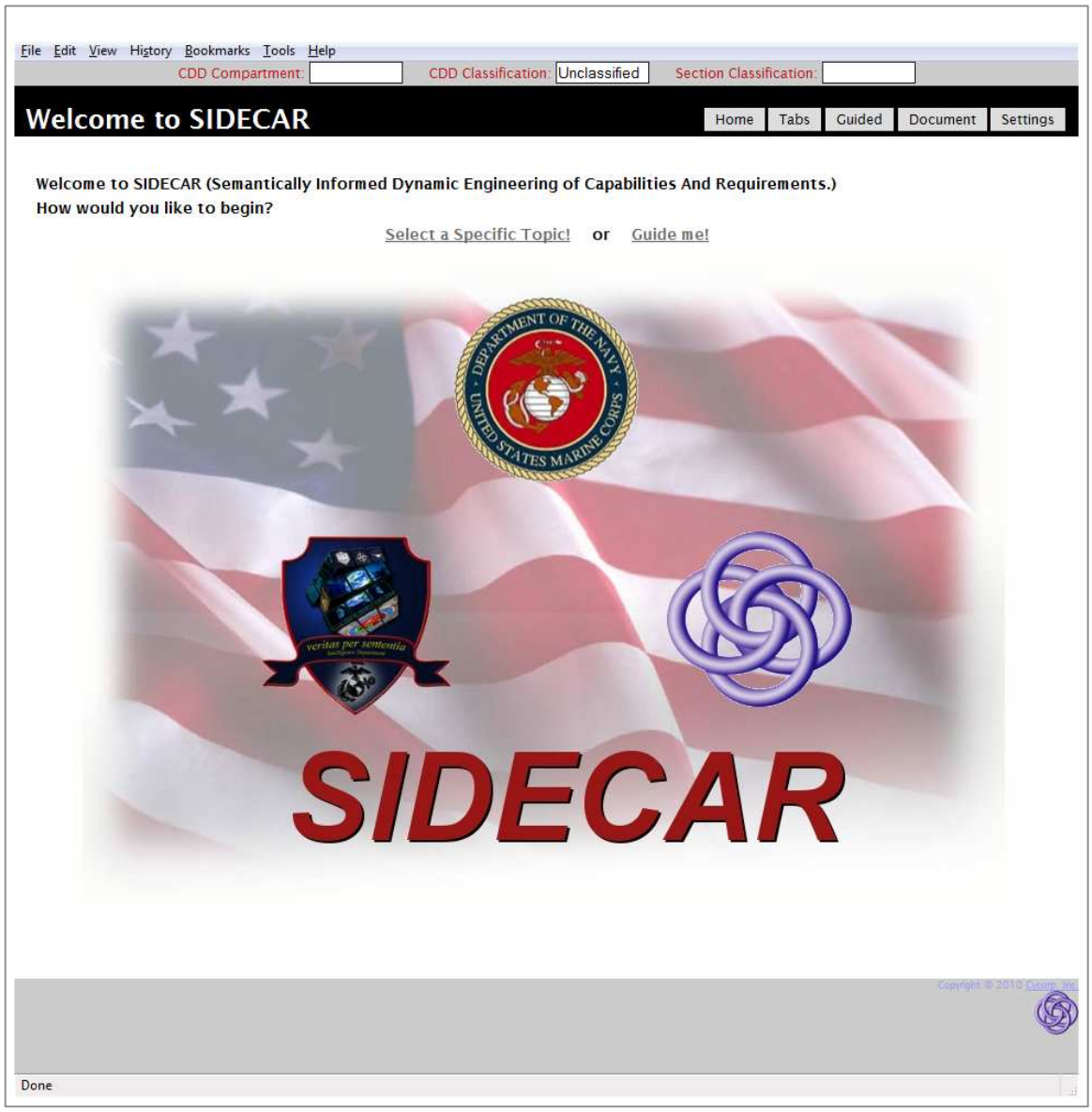


Figure 21. *The SIDECAR system home screen. Buttons linking to the home screen, tabbed and guided views, document view, and system settings appear at upper right. Fields that aid in tracking document compartmentalization and classification appear at top.*

4.2 Guided View

The guided view is analogous to the Turbo-Tax "Guide Me!" conceptual paradigm. The user is led through a linear, line-by-line (or more properly field-by-field) knowledge entry process in preparing the document.

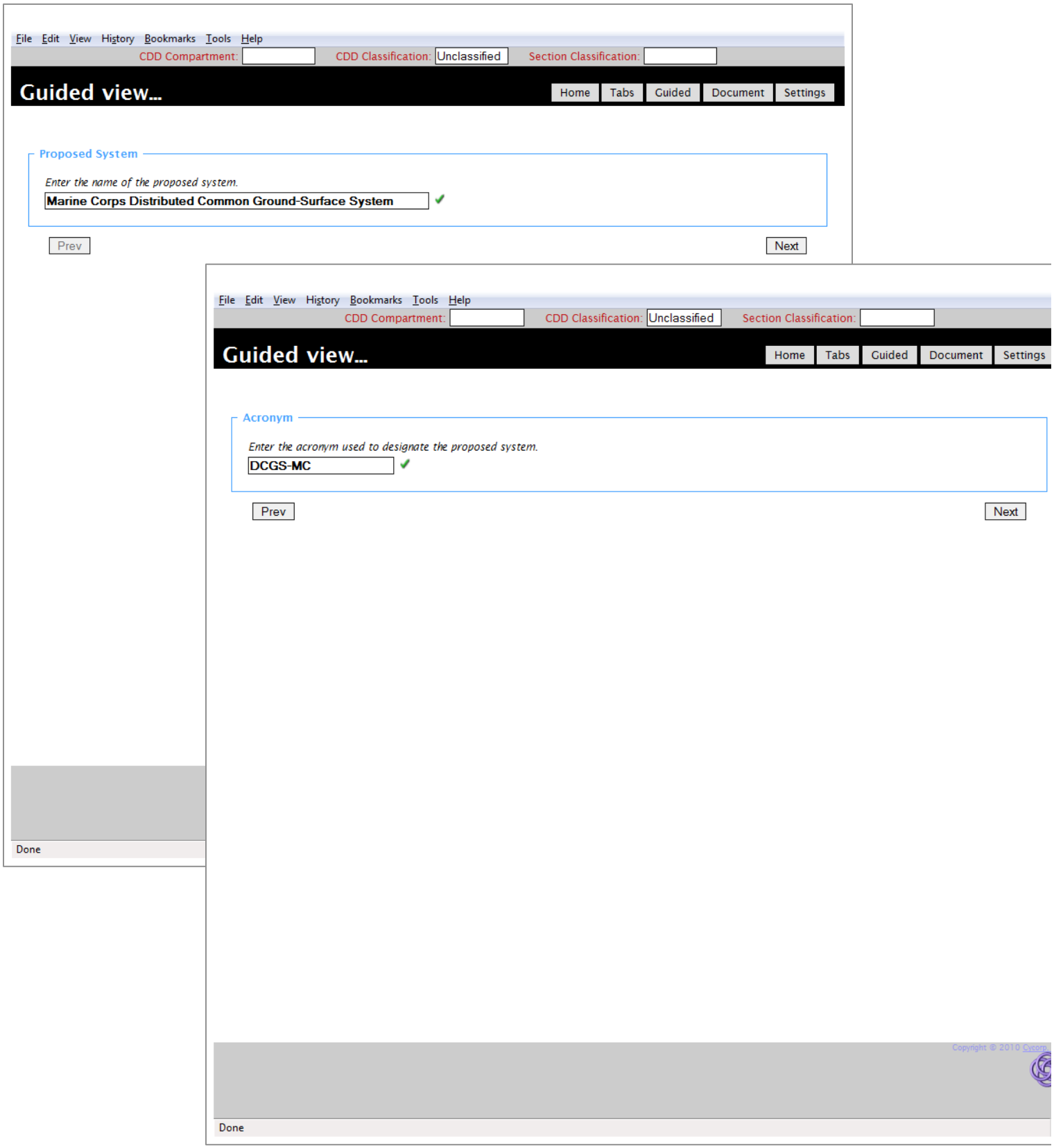


Figure 22. *Two successive screens from the guided sequence, showing the entry of the system name (***Marine Corps Distributed Common Ground-Surface System***) and the subsequent entry of the identifying acronym for the system (* **DCGS-MC***).*

The ‘guided’ view is probably not the recommended modality for entering all of the information in a CDD; however, it is provided as an option for new SIDECAR users or ones who are unsure about some complicated section and who want an absolute, linear order imposed on the data entry for that section.

At any time, the user has the option of swapping between ‘Guided” view and ‘Tabs’ view which organizes knowledge acquisition templates by document section and subsection (in the case of the CDD, tabs correspond to sections, with further organization of templates within the tab based on required subsections.

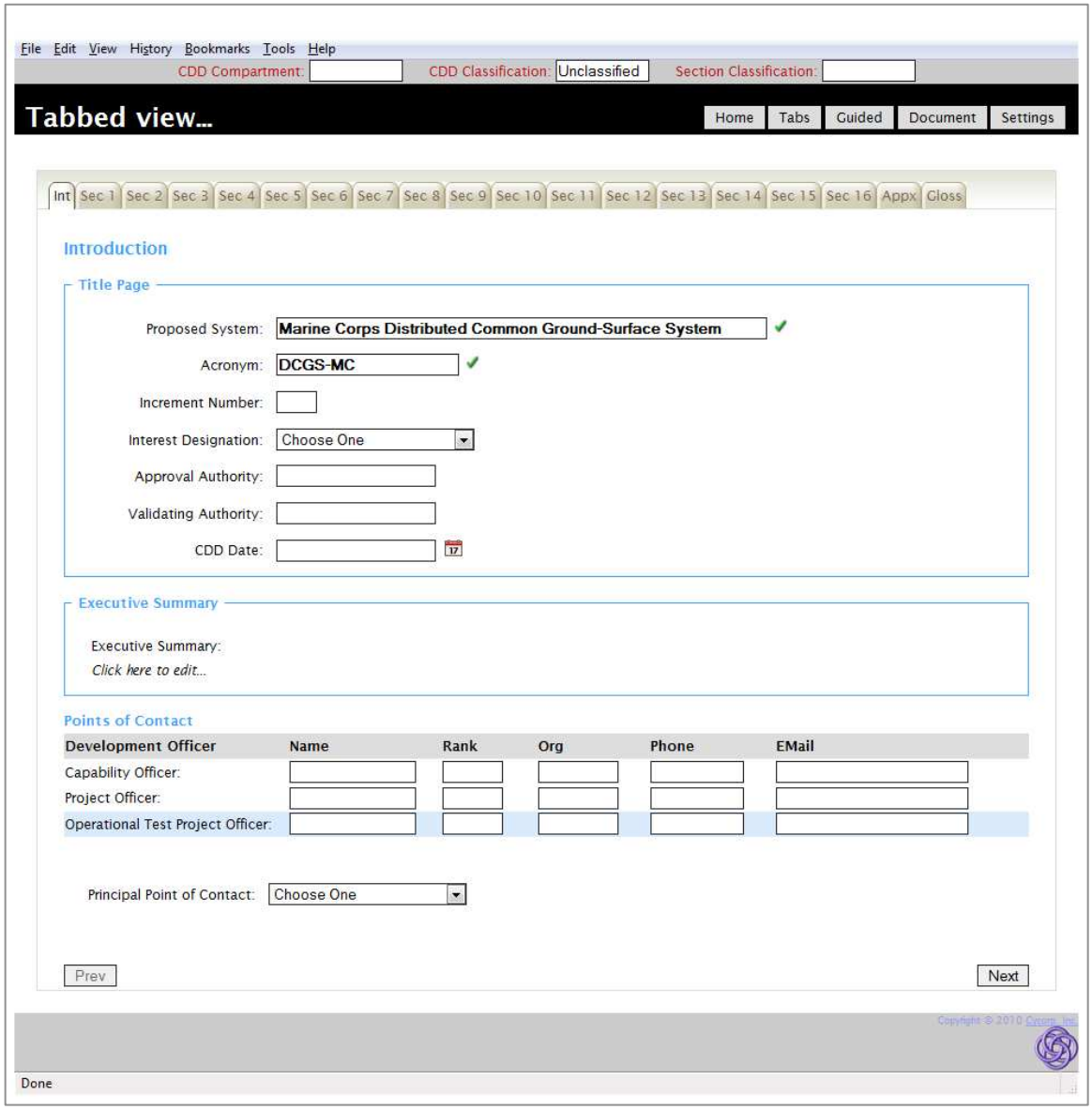


Figure 23. *The ‘Tabs’ view. This particular Tab is the one for the CDD Introduction section; it includes subsections for the title page, executive summary, and points-of-contact.*

4.3 Tabbed View

In figure 23 we see the 'tabbed' view for the Introduction section of the CDD being created by SIDECAR in this run of the system. Sixteen numbered section 'tabs' for the CDD appear at the top, together with tabs for the Introduction, the Appendix, and the Glossary.

Within the tab, templates are further organized by relevant subsection (in the case of the Introduction tab, these subsections include the title page, the executive summary, and the listing of the points-of-contact).

The 'prev' and 'next' buttons are used for navigating between tabs, but, alternatively, the user can mouse click on any section tab to jump directly to that section.

4.4 Document View

A formal logical model of the CDD (and of the state of the SIDECAR session, and of the dynamic actions carried out by the user that led to that state.) The formal model of the CDD can be converted, assertion by assertion, into English, by SIDECAR, by calling on Cyc's English generation subsystem. Each predicate has an NL generation template which enables the system to convert a CycL assertion using that predicate into a more or less readable English sentence. For example, the template for the predicate operationalTestProjectOfficer is:

```
(NPIsXP-NLSentenceFn
  (TermParaphraseFn-NP :ARG2)
  (ConcatenatePhrasesFn
    (BestDetNbarFn-Definite
      (BestNLPhraseOfStringFn "operational test project officer for"))
    (TermParaphraseFn-NP :ARG1)))
```

Given assertions which will be made below, by SIDECAR, along the lines of:

(operationalTestProjectOfficer DCGS-MC EricHollinshead)

and, recursively, for that person's rank, etc., an English sentence can be added to the growing textual CDD, along the lines of:

"Mr. Eric Hollinshead is the operational test project officer for DCGS-MC".

A more complicated example, from later in the session, yields these 2 English sentences:

Any process implementing the DCGS-MC is required to maintain information dispatch. Consequently, computers in the DCGS-MC must exhibit sufficient CPU speed.

At any time, the user can also escape to a current view of the document under construction by clicking on the 'document' button at the upper right-hand corner of the screen. Here, for example, is how the English DCGS-MC CDD document looks once the system name, increment number, and a few other pieces of information have been supplied through the SIDECAR knowledge acquisition interface:

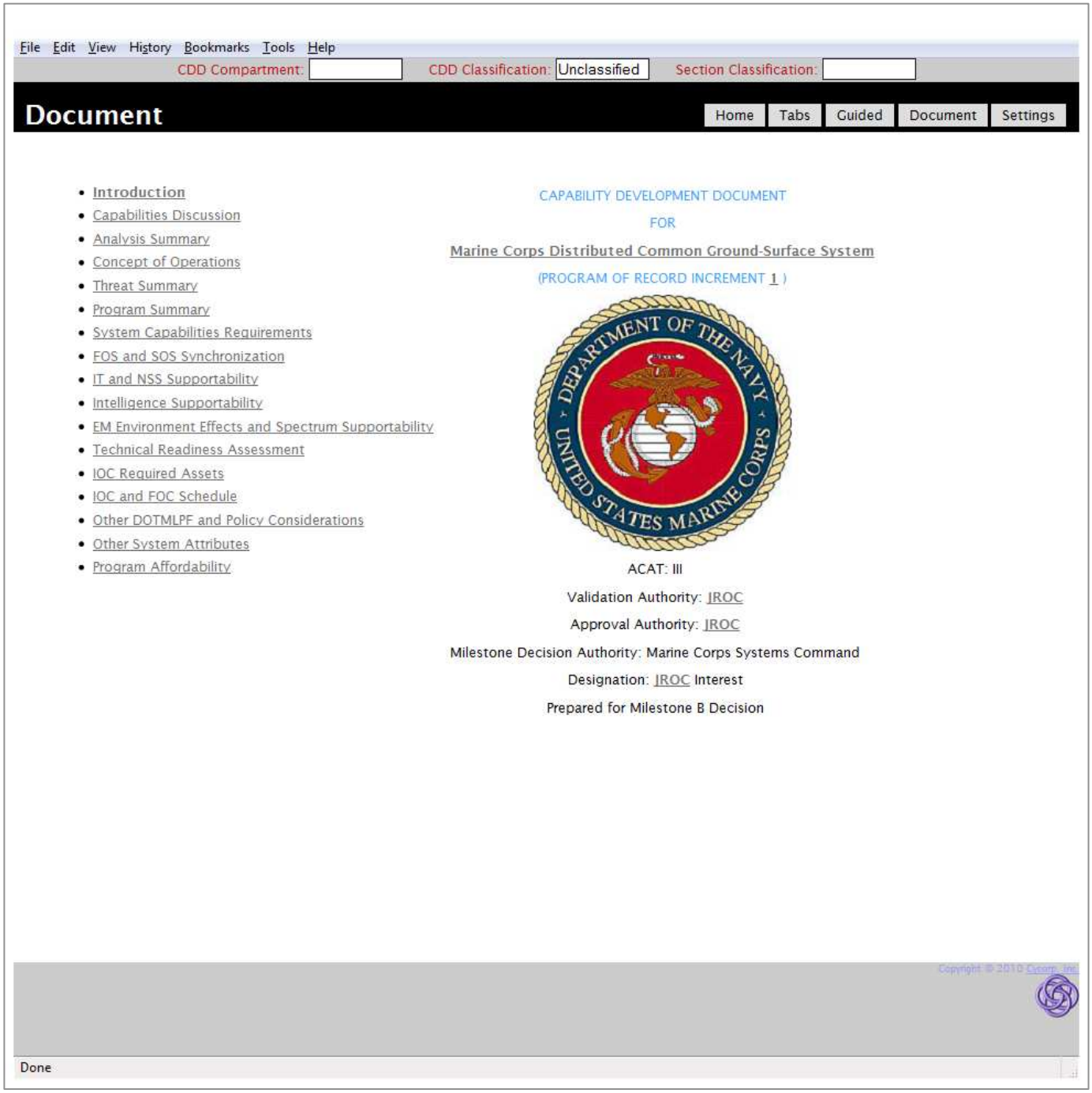


Figure 24. *The English document view, in a very early stage of the interactive run of SIDECAR.*

As another simple example, SIDECAR automatically generates the document title seen in Figure 24 – "**Capability Development Document for Marine Corps Distributed Common Ground-Surface System (Program of Record Increment 1)**" – from the system name and increment number as soon as the relevant information is entered by the user on the "Proposed System" line and the "Increment Number" line.

4.5 Supporting Inferences and Explanations

SIDECAR is more than a passive data capture system for populating a document with terms and sentences. Because the user's entries on KA objects are automatically mapped by natural language understanding to terms that are situated in the Cyc ontology, all of Cyc – everything it knows in its KB, all its reasoning modules, etc. – work together to enable SIDECAR to function as an active intermediary in order to, e.g., flag semantically nonsensical or questionable entries, perform real-time inferences to support further, automated infill in accord with policy, and ask useful follow-on questions in support of further knowledge capture.

Here's a simple example of automated infill support. As a matter of Joint Force policy, there are five legitimate entries that can be made for the *interest designation*: the Joint Requirements Oversight Council (JROC), the Joint Capabilities Board (JCB), Joint Integration, Joint Information, and Independent. The first two are military organizations; the latter three don't specify actual organizations but are rather *classes* that specify the kind of Functional Capability Board that might review the document. But a piece of knowledge (entered for this application) in Cyc's KB is the rule that, when the name of an actual *military* organization is chosen from the interest designation menu, this name should also be entered as the approval authority and also as the validation authority.[19]

If the user selects the string "JROC" or "JCB" from the interest designation selection menu, then, since the knowledge base already knows what those are, namely military organizations, the rule will fire and that same value will automatically be propagated to the approval authority and validation authority fields. More precisely, in the Cyc lexicon there are statements that explicitly list the full names of, and the common acronyms for, these organizations.

These policies, facts, rules, and constraints are represented in the Cyc Knowledge Base; since SIDECAR can invoke them in real time, this saves the user the trouble of having to fill those lines in by hand, saves them from potential mistakes, etc. Moreover, the rules are authored at a *level* that insures that changing small details of the document development environment only requires commensurately small changes in SIDECAR's programming. Using a representation (and a representation language) that is expressive

[19] Cyc also knows – but doesn't use, in this case – the rule that, if the interest designation is a nonmilitary organization, i.e., an FCB class, then, by way of contrast, the approval authority and validation authority must each be a Functional Capability Board.

enough to "impedance match" the application domain is an important element of SIDECAR's success.

Thus, if some new, relevant capabilities review board were to be created that needed to be added to the interest designation candidate set, adding it would be a simple matter of adding its name to the set of permissible selection values, while the underlying rules in the Knowledge Base governing dynamic SIDECAR behavior would not need to change[20]. If one of the menu choices represents an organization which transitions from military to civilian control, then choosing that item on future CDDs Interest Designation menus would *not* automatically fill that in as the Validation Authority or Approval Authority.

The green check-marks reflect filled-in entries (checkmarks) which were *manually* entered by the user. Notice that for the Validation and Approval Authority lines, in figure 25, there is an amber checkmark. The checkmark means that it was unambiguously filled in, but the amber color means that SIDECAR itself inferred that value. Hovering over, or clicking, on a checkmark provides a deeper level of detail: in the case of a green checkmark, this includes who typed the value in, and when. In the case of an amber checkmark, this includes the most salient rule(s) that were used by the inference engine, the assertions which triggered (satisfied) the rules, etc. Figure 22, below, shows what the user sees if they click on the amber checkmark next to the automatically-filled-in JROC entry for the Approval Authority line.

[20] On the other hand, if the *policy* environment were to change, e.g., if it ceased to be the case that a military organization cited as the controlling interest was the default authorizing and validating authority, this more sweeping change would require the supporting rule to be altered – and it could certainly be edited. This is a good rule of thumb: changes to *policies* are likely to require changes to Cyc *rules*.

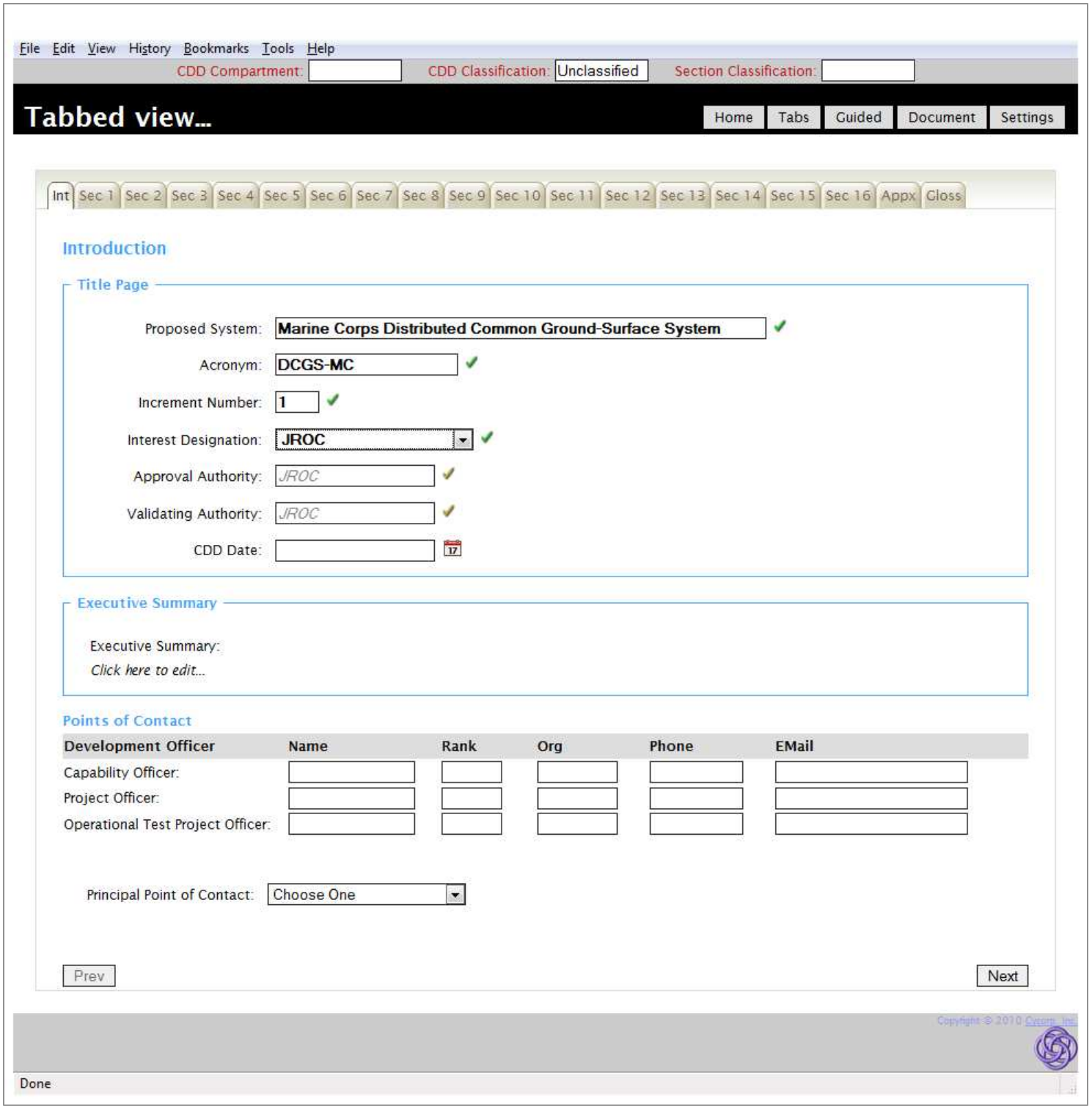


Figure 25. *The introductory subsection with the system name, acronym, increment number, and interest designation filled in by the user, and with the approval and validating authority fields automatically filled in by the Cyc system. The gray font indicates an inference by the SIDECAR system; the amber check marks that appear to the right of the automatically infilled sections can be clicked to obtain explanation; see Figure 26, below, for an example of that.*

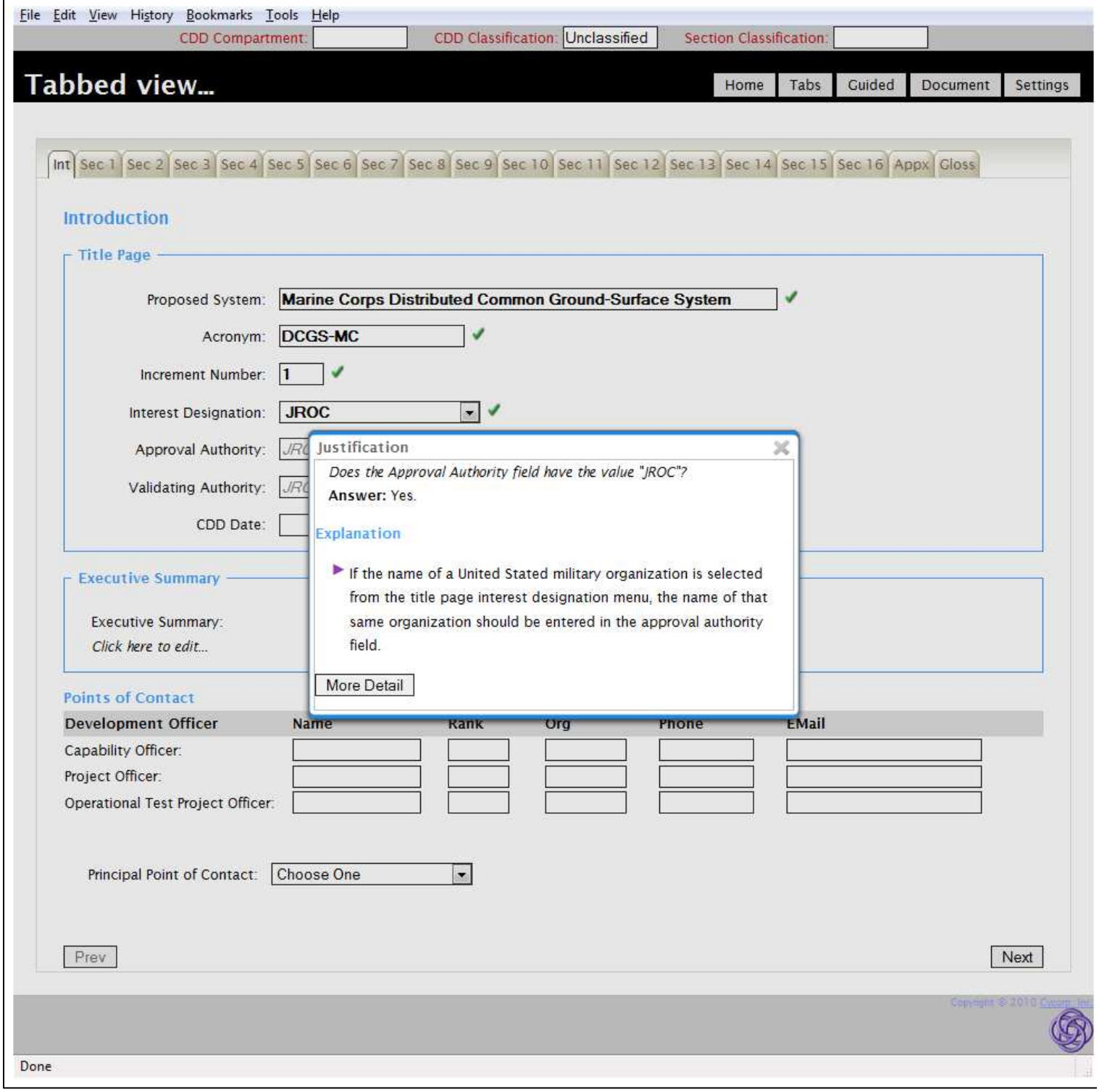


Figure 26. *Justification for the automated infill of the approval authority field. Note that the user can request increasingly more and more detail in the justification; for an example of that, see Figure 27.*

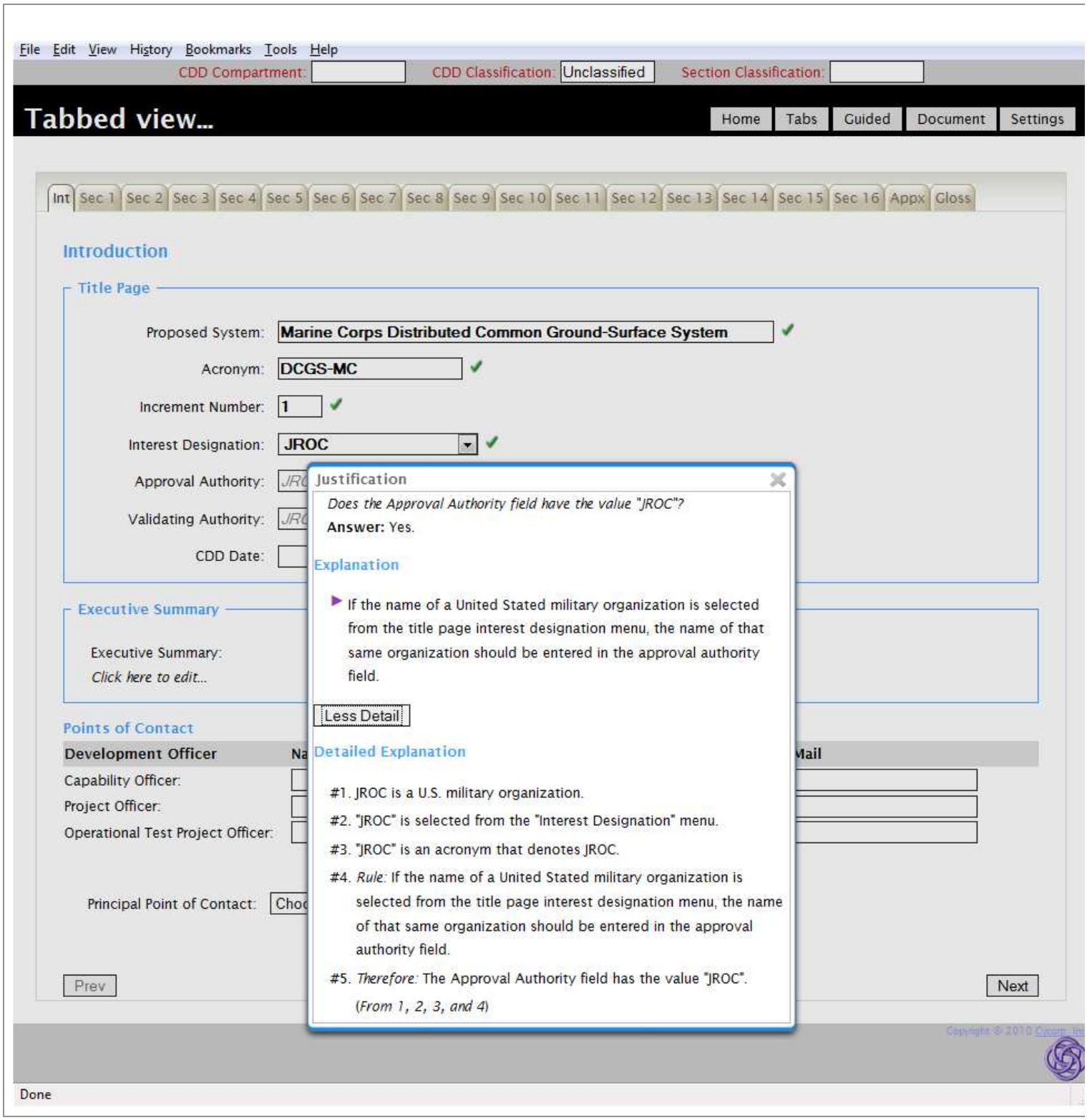


Figure 27. *Increasingly more detailed justification (of the reasoning behind the JROC entry filled in as the approval authority) can be supplied, by SIDECAR, if the user asks for it.*

4.6 Follow-Up From Free Text Entry: The CDD's Executive Summary

Although much of the fact entry in the SIDECAR prototype is handled through structured templates, use of free text entry – unrestricted natural language typed in by the user – is permitted under 'controlled conditions'. A good example of this is the paragraph-sized box in which the user is asked to type in the Executive Summary for the CDD.

Complete understanding of arbitrary unstructured text is well beyond the capability of any contemporary AI system, but Cyc is capable of *partial* parsing and understanding – for example, recognizing various concepts within a body of text. Cyc can then use that partial understanding to drive a clarification dialogue to understand (at the CycL level) more and more of the content of that English paragraph. This is the basis for the Cleveland Clinic application of Cyc described in [Lenat *et al* 2010].

For example, a rule in the Cyc knowledge base says that if Cyc recognizes a military organization type (for example, Marine Air-Ground Task Force) in the executive summary, one good follow-up question to ask would be whether operations characteristic of that organization (e.g., MAGTF operations) or type of organization are supported by the proposed system.

As shown in Figure 28, the user can -- as with all actions that SIDECAR undertakes 'autonomously' -- ask for a justification of its actions.

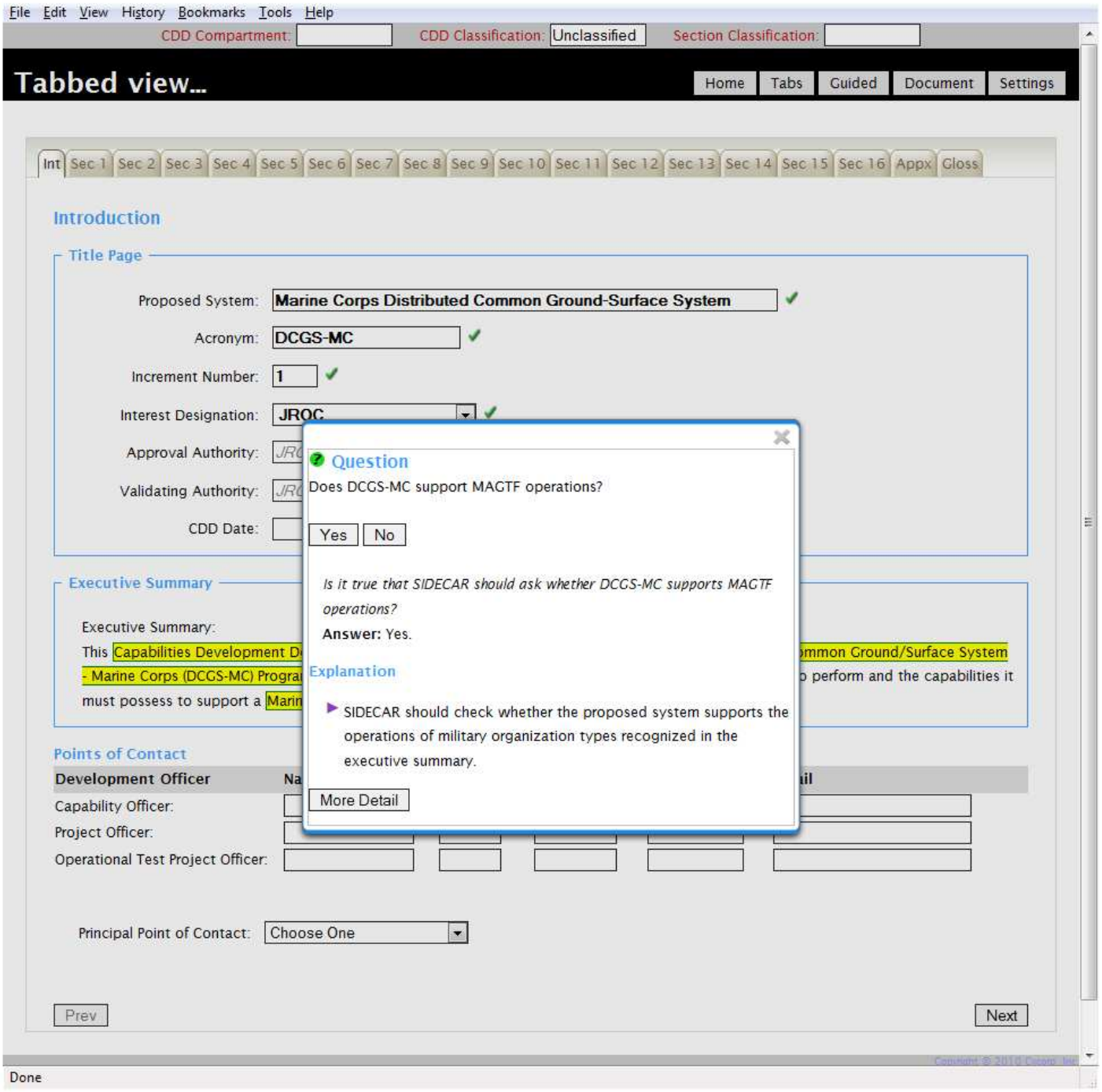


Figure 28. *Yes-or-no question submitted to the user. SIDECAR asks whether or not the DCGS-MC supports Marine Air-Ground Task Force operations, as the result of having successfully recognized and understood the acronym 'MAGTF' in the executive summary entry. Again, the user can obtain an explanation of why the question was generated.*

4.7 Follow-Up From Structured Entry: Principal Points of Contact

More often, SIDECAR's follow-up questions will be triggered by entry through structured templates. For example, in building the current SIDECAR application, we told Cyc a rule that says that, if any development officer of a proposed system is a member of an organization that has a controlling interest in development of a Joint Force Enterprise Project (call it X), then SIDECAR should hypothesize (and inquire of the user) about whether the system now being proposed and having its CDD worked on is subsumed by X – in effect, is a means to accomplishing some of X's required functionalities.

In the current session, for example, the operational test project officer of DCGS-MC has been entered as Eric Hollinshead, and we know that he is a member of the Marine Corp Operations Testing and Evaluation Activity organization (MCOTEA), which has a controlling interest in development of the Marine Corps Intelligence, Surveillance, and Reconnaissanace Enterprise (MCISR-E) [United States Marine Corps, July 2007].

SIDECAR recognizes 'MCOTEA' as the acronym of the testing agency, and the Cyc Knowledge Base includes the information that the person whose name is entered in the operational test project officer field is a member of the organization whose name is entered in the corresponding organization field.

This information -- plus the aforesaid rule -- gives rise to the SIDECAR question that pops up at that point, to the user, about whether MCISR-E subsumes the proposed system. See Figure 29.

In fact the answer is Yes, SIDECAR is subsumed by, and satisfying some of the requirements of, MCISR-E. This in turns causes SIDECAR to want the user to turn their attention to Tab 1: Capabilities; note how the "Sec. 1" tab near the top of the screen is highlighting in green now, in Figure 30, once the user says Yes to the question that popped up in Figure 29.

The user is free to ignore that and go whereever they choose, next, but the guidance is there in case they want to follow it.

Another thing which pops up at this point is a warning (a yellow highway-warning-sign like icon) which the user can address or defer. See Figure 30.

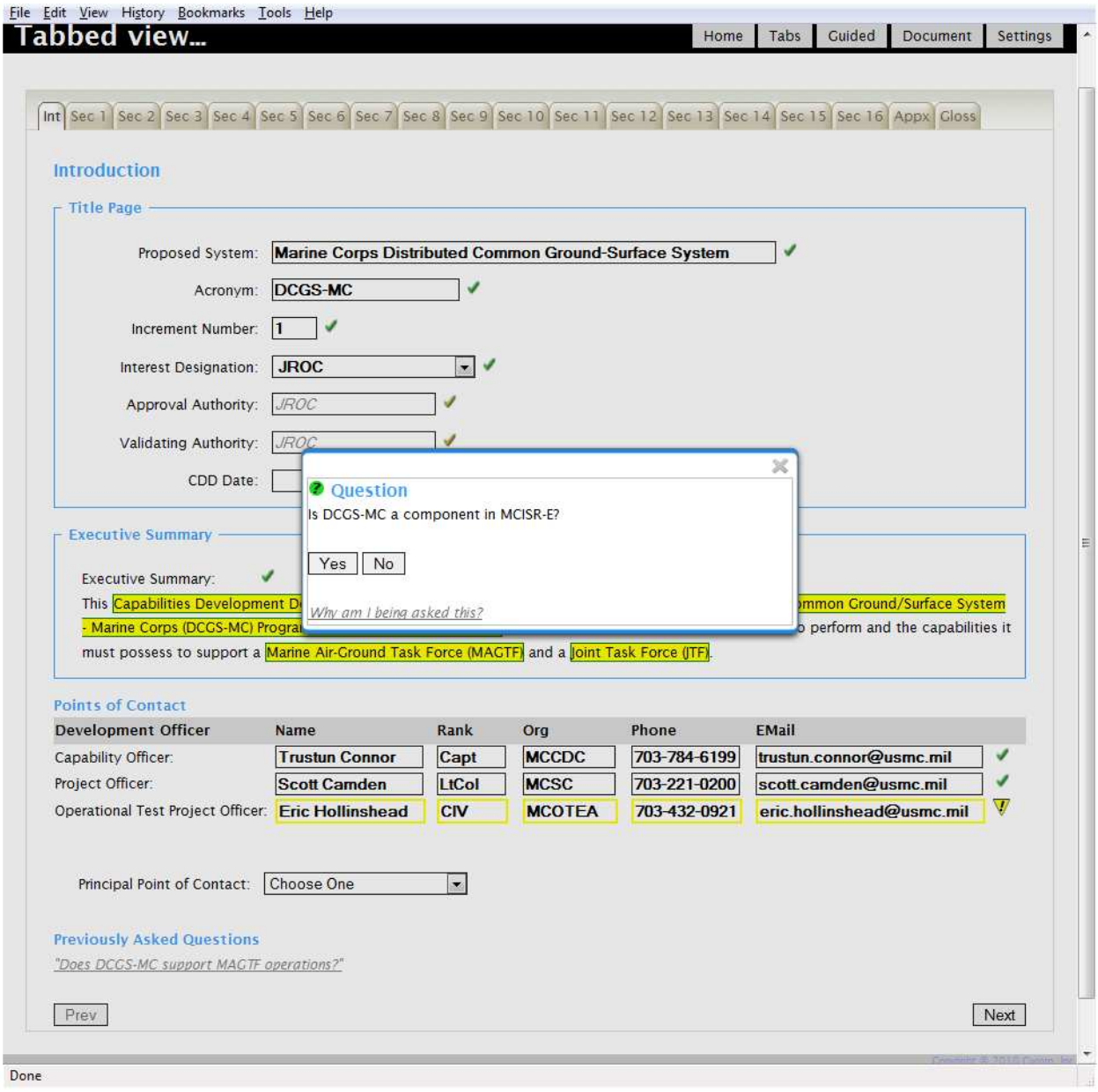


Figure 29. *Another yes-or-no question for the user, deriving from the fact that a development officer is identified as a member of the Marine Corps Operational Test and Evaluation Activity organization, which has an interest in the development enterprise of a known system.*

4.8 Anomaly Detection

As was mentioned above, SIDECAR can also flag potentially anomalous or problematic user behavior. This can range all the way from simple recognition-based type checking (e.g., flagging an alphabetic character entered in the increment field, or an ICD entered in a list of terms which are all expected to be MUAs) to complex real-world anomaly detection. In this case, one of the individuals identified as a point-of-contact in the POCs section is a civilian. Cyc, and hence, SIDECAR, has the information that development officers for proposed Joint Force systems are usually active duty military officers; the user can ignore this alert, respond to it, or defer it until later.

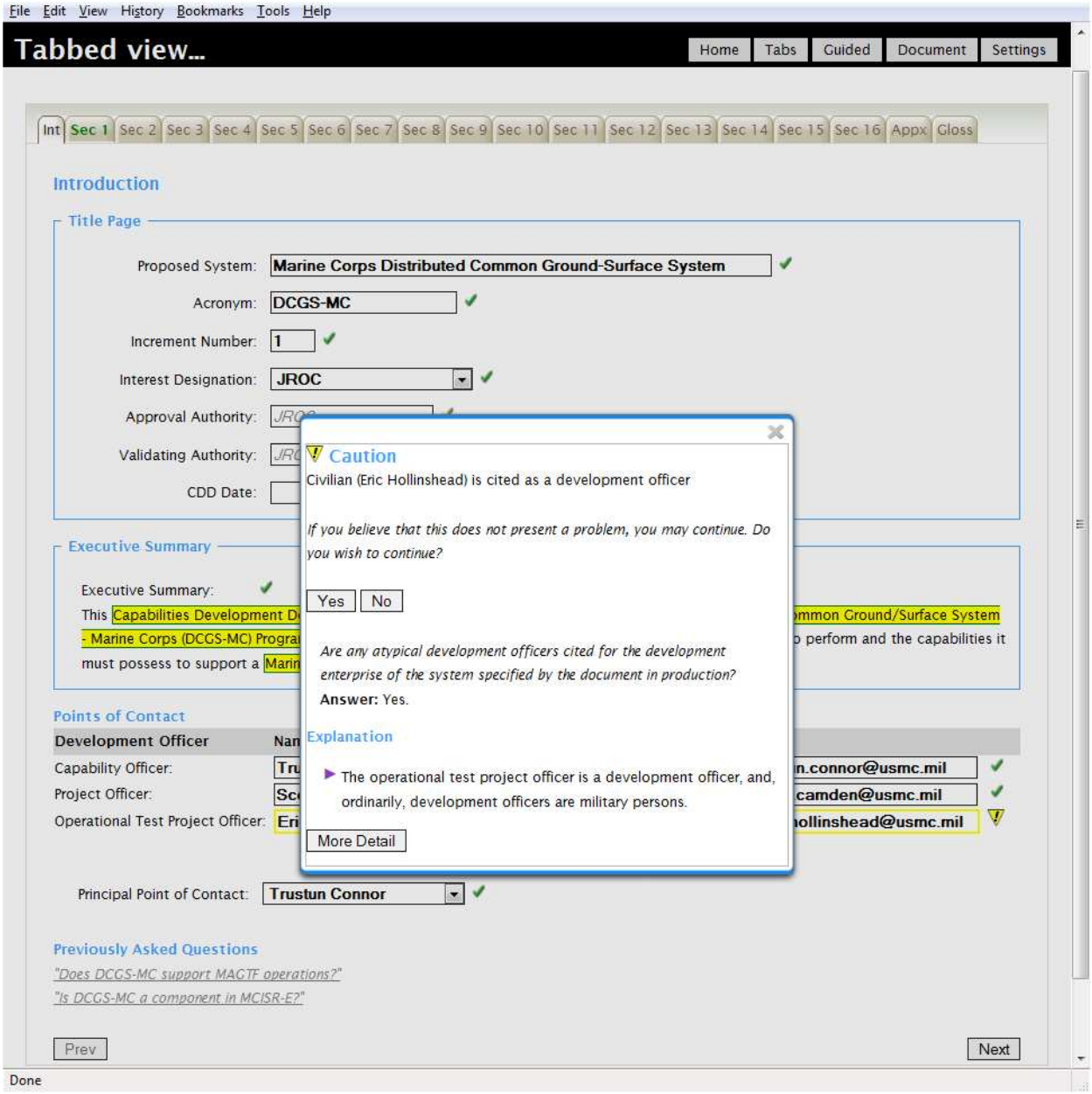


Figure 30. *An example of anomaly detection in SIDECAR: a caution flag has been presented to the user because Eric Hollinshead, one of the identified development officers, is a civilian, which is an atypical attribute for a development officer. Note also that the tab for the CDD section 1 (the Capabilities Discussion) is now showing highlighted in green font (upper left-hand corner of the screen). This is SIDECAR's way of unobtrusively letting the user know the next recommended section to address.*

4.9 Other Ways of Querying SIDECAR-Derived Knowledge: the CAE

As additional information is entered, the content accessible in the synthesized English version of the document view grows. See Figure 31:

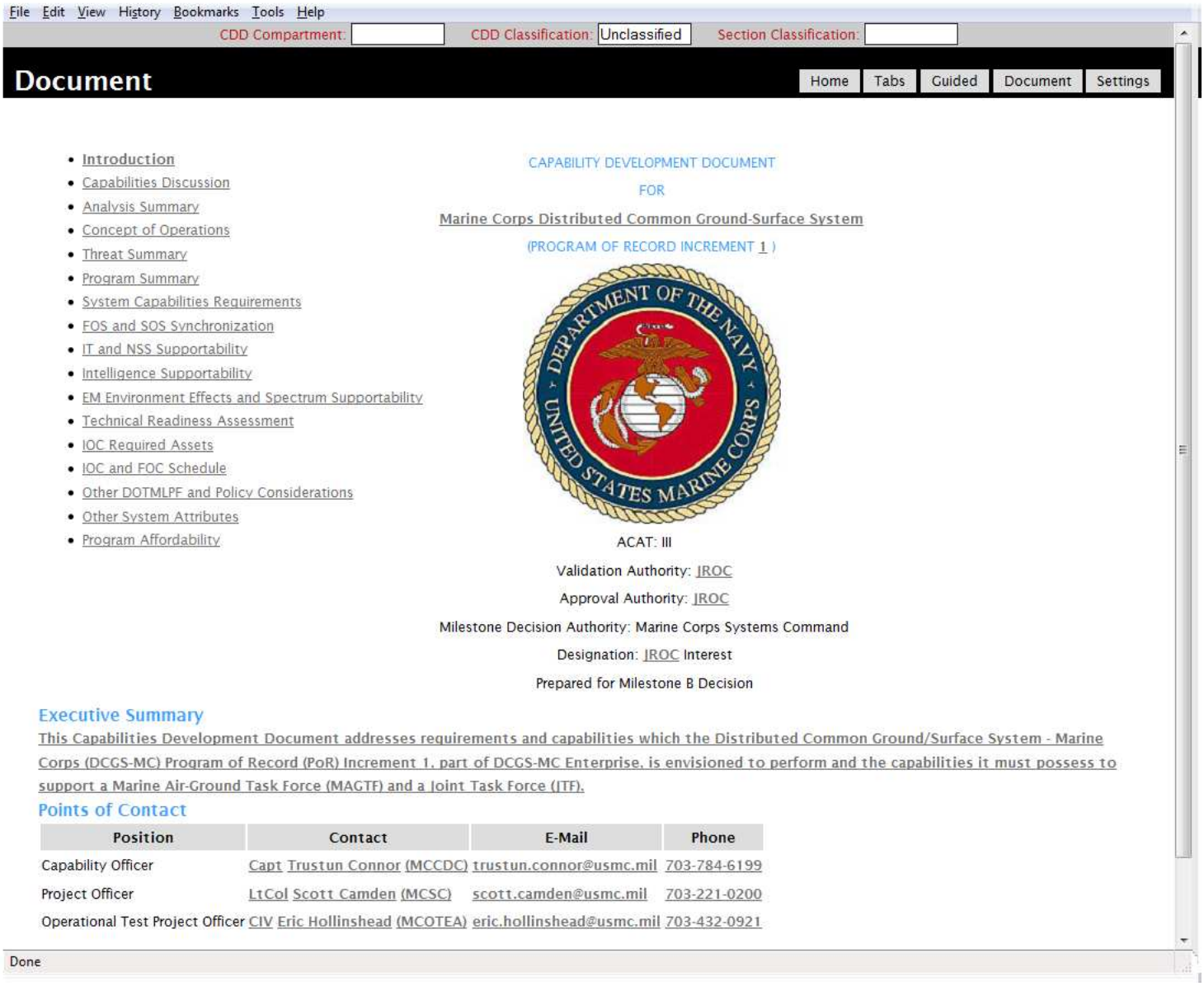


Figure 31. *The document view, with the introduction section tab completed.*

It is important to bear in mind, though, that this information is stored not only in the document, but in a formal model in the Knowledge Base which can be combined with background knowledge and mined for additional information.

To explore what we mean by this, we can invoke a query system called CAE for "Cyc Analytic Environment" developed at Cycorp over the course of several government and

commercial projects. CAE allows lightly trained users to formulate and ask queries that can be run against the Knowledge Base. [Lenat *et al* 2010] Below, in Figure 32, we see a CAE screen displaying a set of formulated and saved queries in English form.

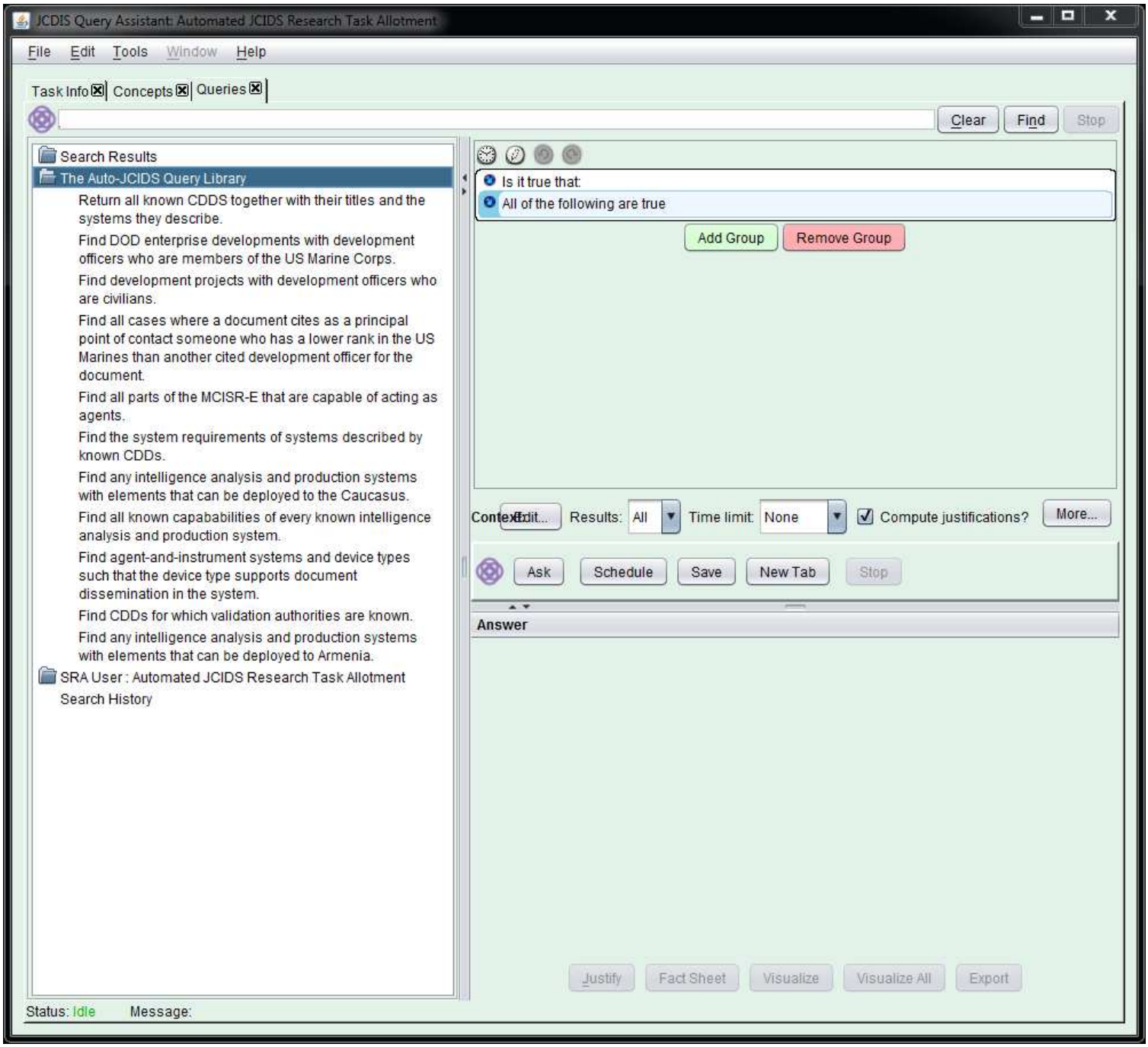


Figure 32. *The Cyc Analyst's Environment query interface. English specifications of saved queries appear on the left. If a query is selected, a pseudo-English rendering that is a closer approximation of the CycL form will appear in the upper right-hand pane (the user also has the option of viewing the CycL form of the query. The query is asked using the controls in the middle of the right-hand side of the interface.*

One of these asks for systems with development officers such that the *principal* point of contact is outranked by another other development officers. This information isn't *explicitly* provided by any facts entered through SIDECAR, but if Trustun Connor is selected as the PPOC, it is in fact the case (Scott Camden, a Lieutenant Colonel, outranks Trustun Connor, a Captain). If we ask this query in the CAE, we get Trustun, Scott, and their respective ranks as bindings, because the Knowledge Base includes the knowledge that Captain is a 'following value' of Major, which in turn is a following value of Lieutenant Colonel on the Marine Corps ranking scale.

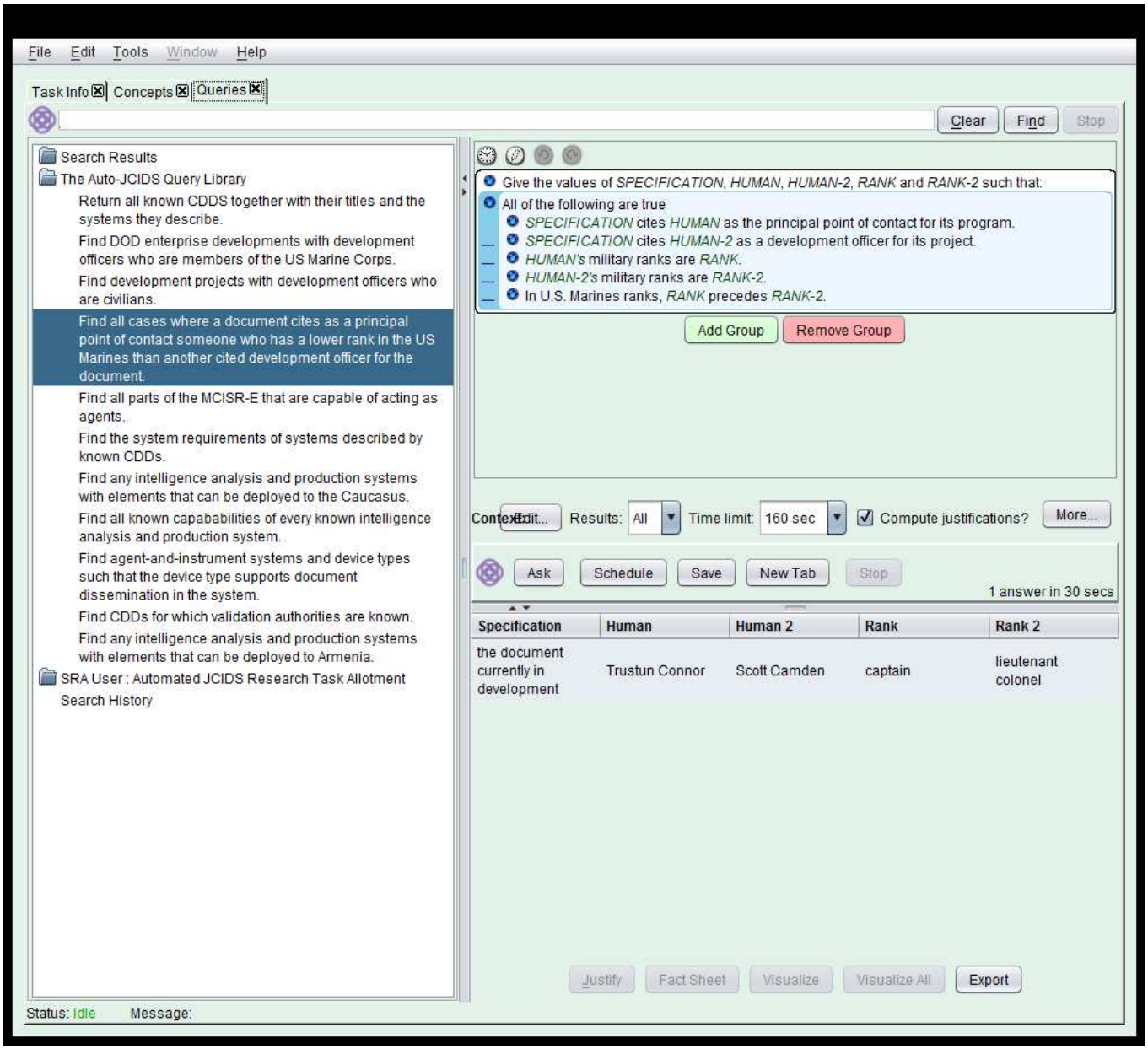


Figure 33. *The result of searching for documents where the cited principal point of contact is outranked by one of the other development officers. Results of search are shown in the lower right-hand pane of the user interface.*

The preservation of the formal system model and the ability to pose such queries against it are deemed to be among the most important features of SIDECAR in that they contribute pre-eminently to the system's extensibility, flexibility, and overall utility. Among other options, it allows the user to investigate interactions of the system being described with other known systems, in addition to allowing for explication of the consequences of design choices made with respect to the system under development – a feature which could be important for system architecting, as we shall see.

4.10 Interactive Diagrams as a mode of entering new facts

One of the fact entry modalities that we are currently exploring with SIDECAR is interactive diagrams. Under certain circumstances, a graphic (node-and-link) diagram may be a far better mode of presentation than a hierarchical menu for presenting a user with options, and in the limit, such a diagram can function as a direct modality for the user to manipulate the architecture of the underlying model (changing links, redirecting links, eliminating or adding nodes, etc.). Below is a very simple interactive diagram that the SIDECAR system presents for the FOS/SOS Solution Section 6 of the CDD template.

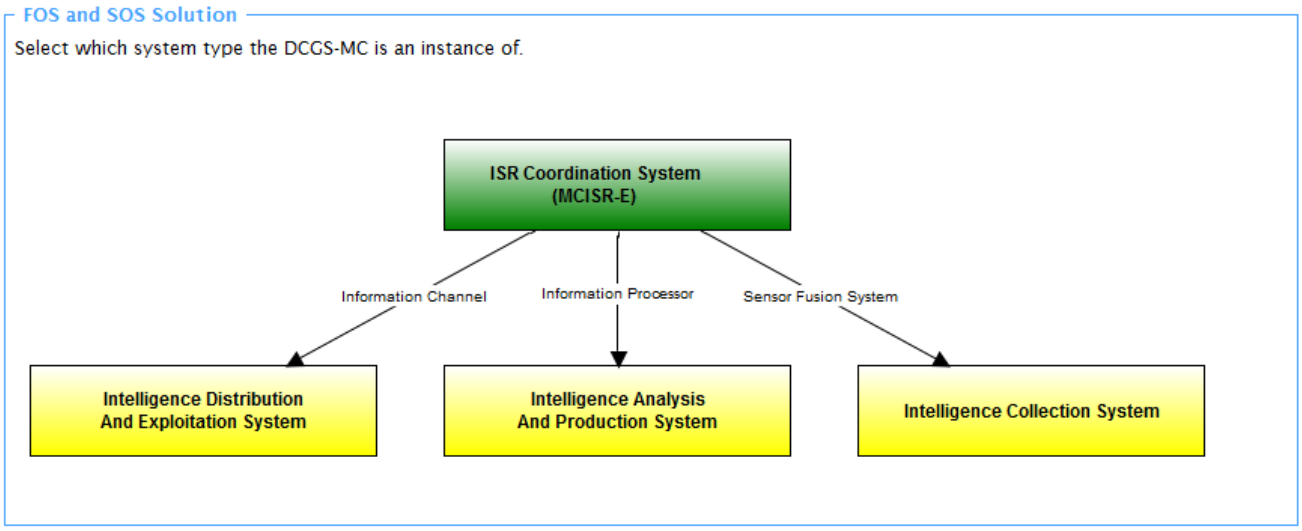


Figure 34. *Simple tree-structure graphic from the FOS/SOS (families of systems / systems of systems) solution subsection, showing the functional categories associated with an ISR coordination system (the type of the Marine Corps ISR Enterprise System). The intent is to have the user select the functional role which best fits the DCGS-MC as a subcomponent of this system.*

We generate this diagram via a rule that says that if the parent system of the proposed system of the CDD is a type for which 'functional role requirements' are known (requirements that specify a subsystem type and a systemic role) these requirements should be used to generate a diagram, and the user should be asked which subsystem type, if any, the system being described instantiates. Recall that, above, SIDECAR asked the user whether the MCISR-E was the parent system of the DCGS-MC in a follow-up

query. The user's affirmative response – or, rather, the assertion that Cyc infers, based on this response – is used as the support for generating this diagram. The Knowledge Base already contains the information that MCISR-E is an ISR (Intelligence, Surveillance, and Reconnaissance) coordination system, and the information that such a system has at least three, top-level functional role requirements: it needs an intelligence collection system to play the role of sensor integrator; it needs an intelligence analysis and production system to play the role of information processor; and it needs an intelligence distribution and exploitation system to play the role of information channel - roles that in fact correspond to the three operational 'pillars' of MCISR-E, filled by Persistant Intelligence Surveillance and Reconnaissance (P-ISR), the DCGS-MC, and Intelligence Dissemination and Utilization (IDU, formerly Actionable Intelligence) [United States Marine Corps 2010, pp. 1-2 (Capabilities Discussion)]. In the diagram, the user is being asked which of these system-type-and-role combinations, if any, best describes the role of the DCGS-MC within the MCISR-E architecture. The correct choice, from the standpoint of the DCGS-MC's actual function within the MCISR-E ConOps, is 'Intelligence Analysis and Production System'. Selecting this box (by clicking on it), will result not only in the inference, based on the user action, that the DCGS-MC is an intelligence analysis and production system, but also in the inference that it plays the role of information processor with respect to MCISR-E.

4.11 Menu Dependency Example: Capability Analysis

Situating the DCGS-MC within the MCSIR-E ConOps architecture can serve in turn as the starting point for the Capabilities Discussion. Appendix 'A' of Enclosure 'G' of the July 2009 version of the JCIDS operations manual requires that this section explicate the relationship between the capabilities of the system being described and the capability of the parent system: more specifically, that it explain how the system requirements occasioned by the overall role and objective of the proposed system within its operational context (as established by the FOS/SOS solution) will be implemented, with particular focus on required infrastructure capabilities and the hardware used to implement them. [United States Department of Defense, July 2009, p. 91] Essentially, the section requirements call for a *functional analysis* comprised, not just of a set of statements from the user describing the physical infrastructure of the system, but of explanations of the capabilities afforded by this infrastructure, and of the role-engendered requirements these infrastructure capabilities satisfy.

Comparison of existing CDDs[21] tends to suggest that the discussion section historically suffers from the radical diversity of mental models that authors bring to the table when interpreting terms like 'capability', 'infrastructre', and 'requirement' – which suggests

[21] Compare, for example, the extant DCGS-MC CDD (United States Marine Corps, 15 January 2010) and the Communications Emitter Sensing and Attacking System (CESAS) CDD (United States Marine Corps. *Capability Development Document for Communications Emitter Sensing and Attacking System Increment 1*. 7 July 2006.)

that a very important innovation SIDECAR can introduce in the development process is the use of formal systemic models to drive knowledge acquisition for such discussions.

The demo provides a notional use case of how this might be done through menu dependencies. Underneath the FOS/SOS Solution diagram in section 1 are three menus named, respectively, 'System Requirements', 'Infrastructure Capabilities', and 'Implemented Via'.

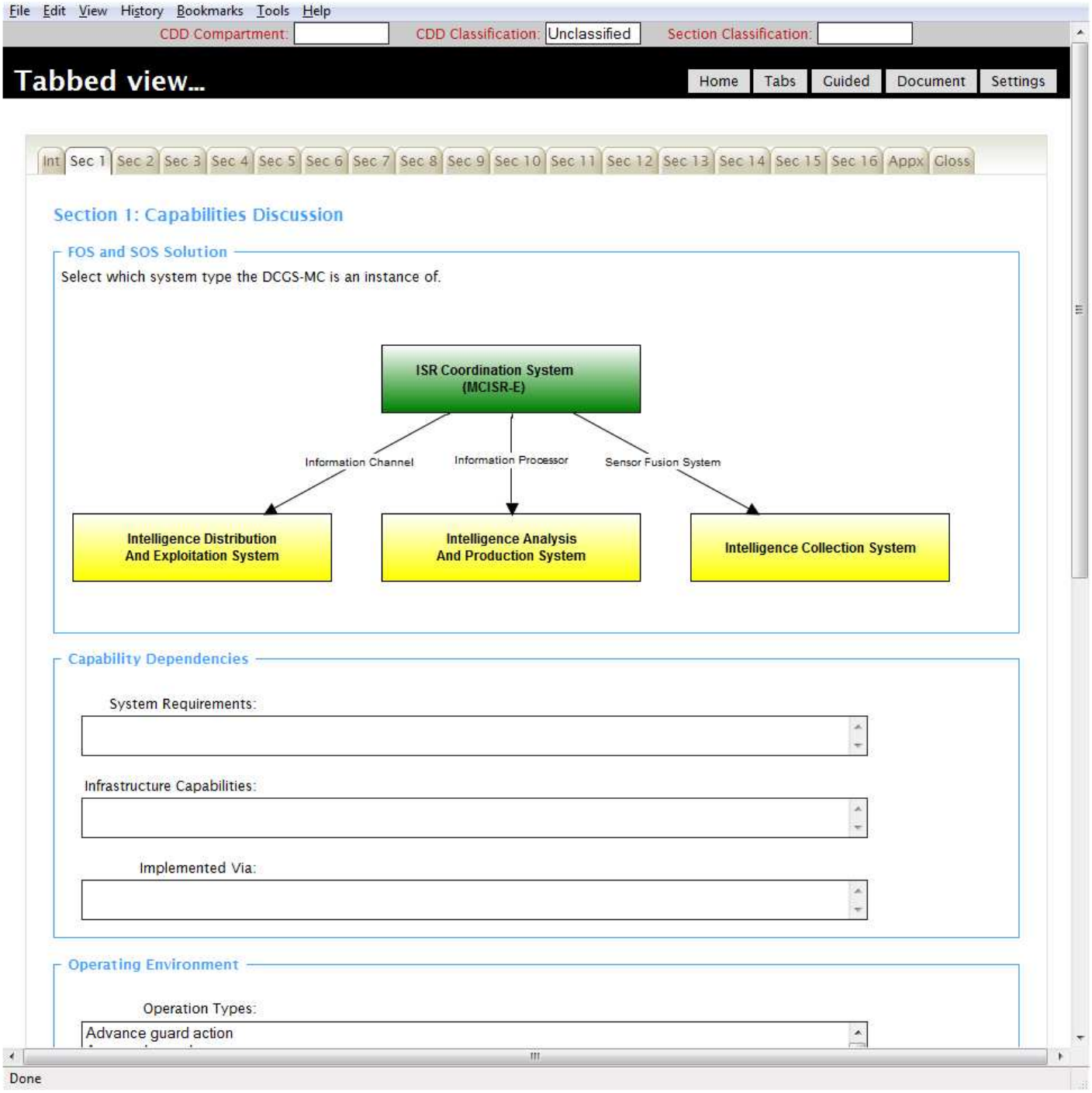
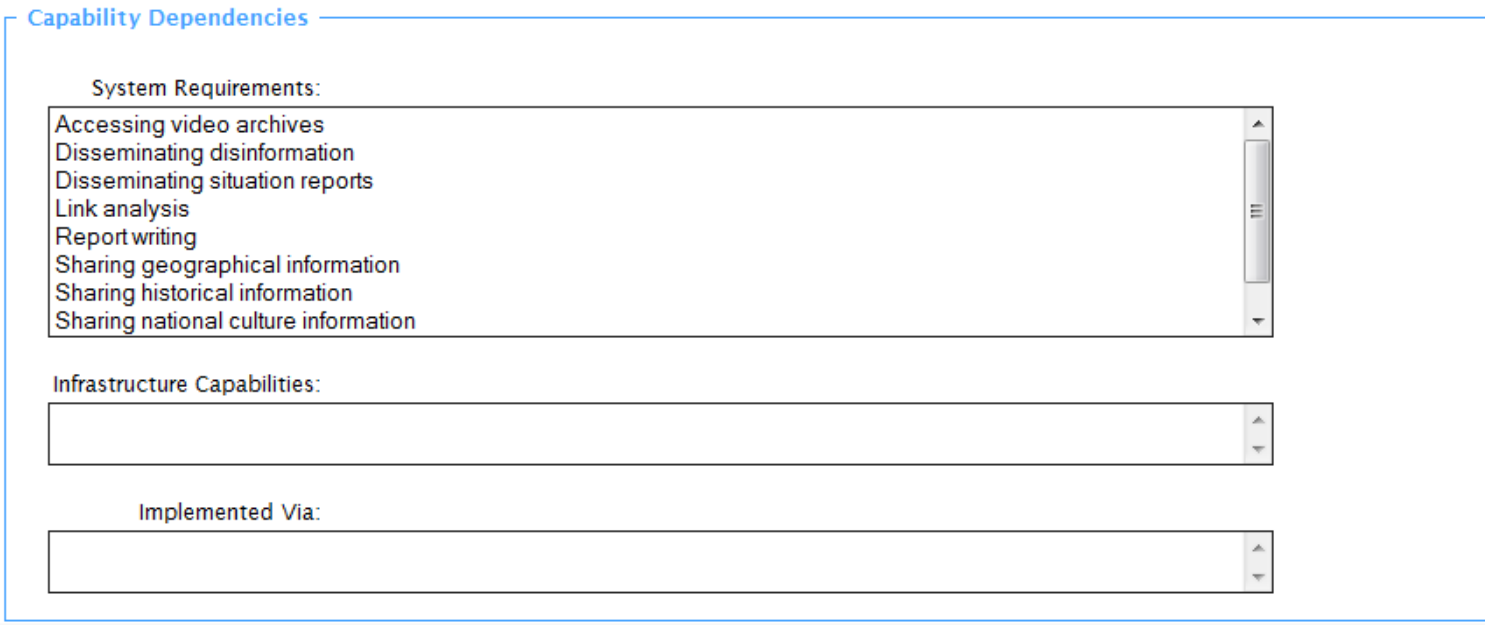


Figure 35. *The Capability Discussion menus as they appear under the FOS/SOS solution diagram.*

As soon as a functional role is identified for the DCGS-MC from the FOS/SOS solution diagram, the 'System Requirements' menu populates with performance requirements



inferred from the fact that the DCGS-MC has this role (in the present case, the role of information processor *qua* Intell. Analysis and Production System in the MCSIR-E).

Figure 36. *The 'System Requirements' menu population that results from selecting the 'Intelligence Analysis and Production System' role from the FOS/SOS solution architecture.*

Selecting one or more of these will result in the 'Infrastructure Capabilities' menu populating with action types that the Cyc system can infer any system capable of performing the selected system requirement(s) will need some kind of device to play. E.g., if 'disseminating situation reports' is selected, Cyc can infer that this activity presupposes device infrastructure capable of creating situation reports, transmitting information from one communication device to another, and, propagating information through computer networks.

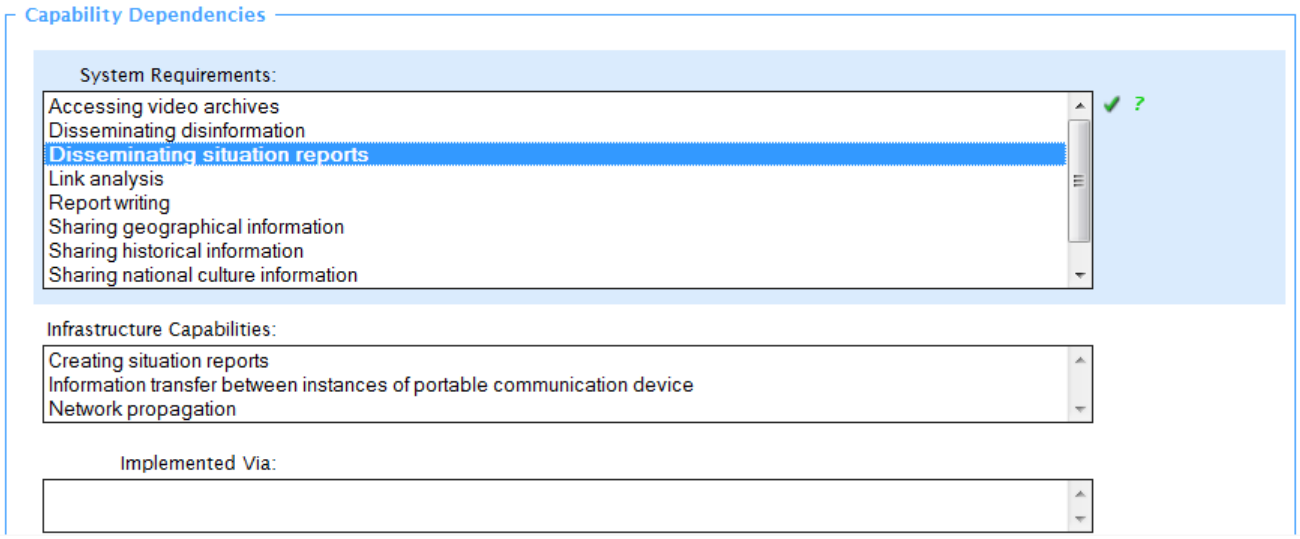


Figure 37. *The'Infrastructure Capabilities' menu population that results from selecting 'Disseminating situation reports' from the 'System Requirements' menu.*

Selecting one or more infrastructure capabilities in turn will lead to the 'Implemented Via' menu populating with devices and device types that are inferred to be capable of implementing the selected infrastructure capability. Thus, for example, selecting 'Network propagation' will result in 'Implemented Via' populating with, among other things, various kinds of computer network:

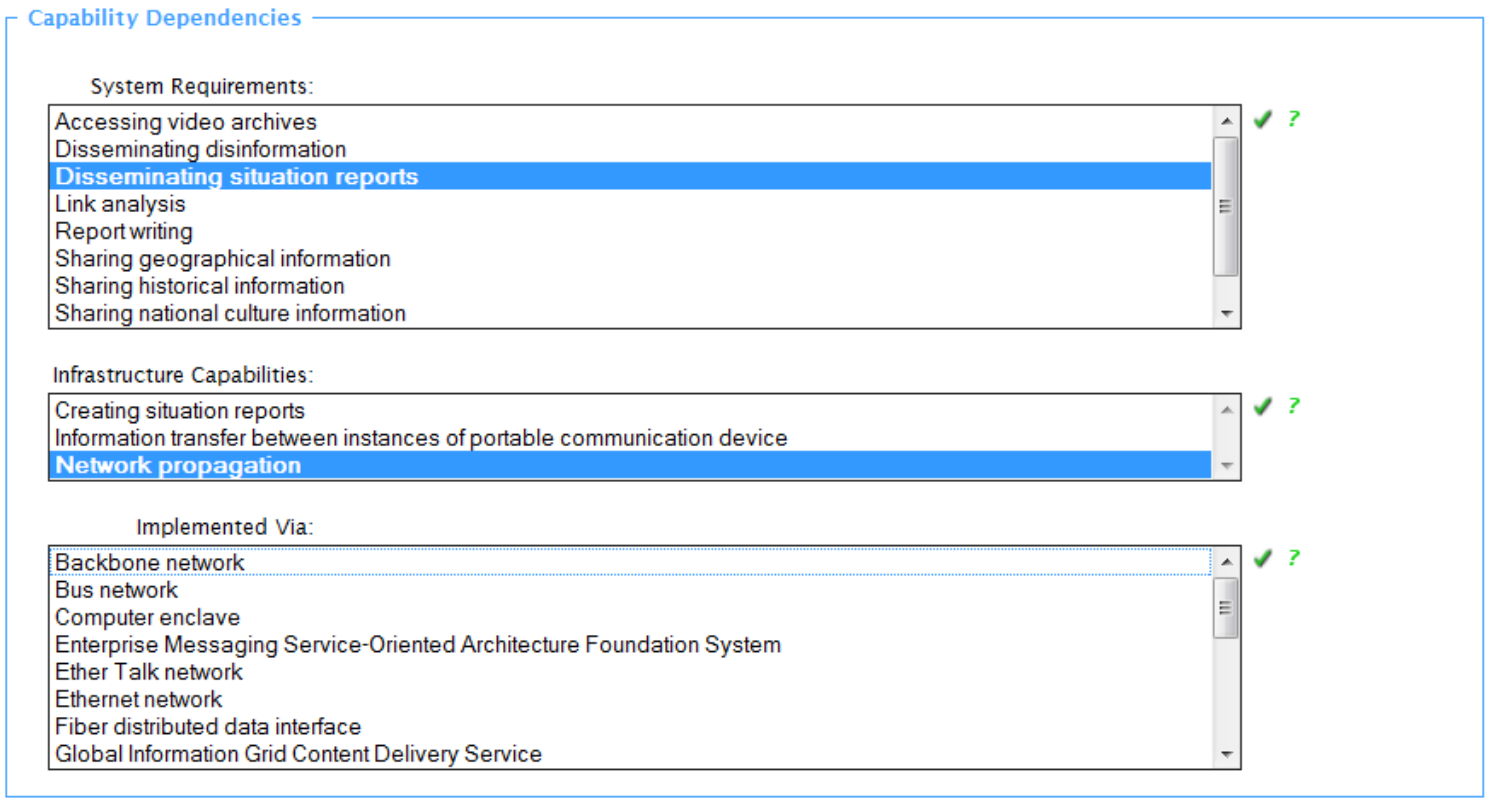


Figure 38. *Here we see the 'Implemented Via' menu population deriving from a selection of 'Network Propagation' in the 'Infrastructure Capabilities' menu.*

The menus are defined in such a way that additional selections in one menu can determine additional selections in another. For example, suppose that, in addition to 'Disseminating situation reports', 'Disseminating disinformation' is selected as a system requirement. We now see, in the 'Implemented Via' menu, a number of network types that we did not see there before – specifically, known network *vulnerabilities* that might be exploited in disseminating disinformation.

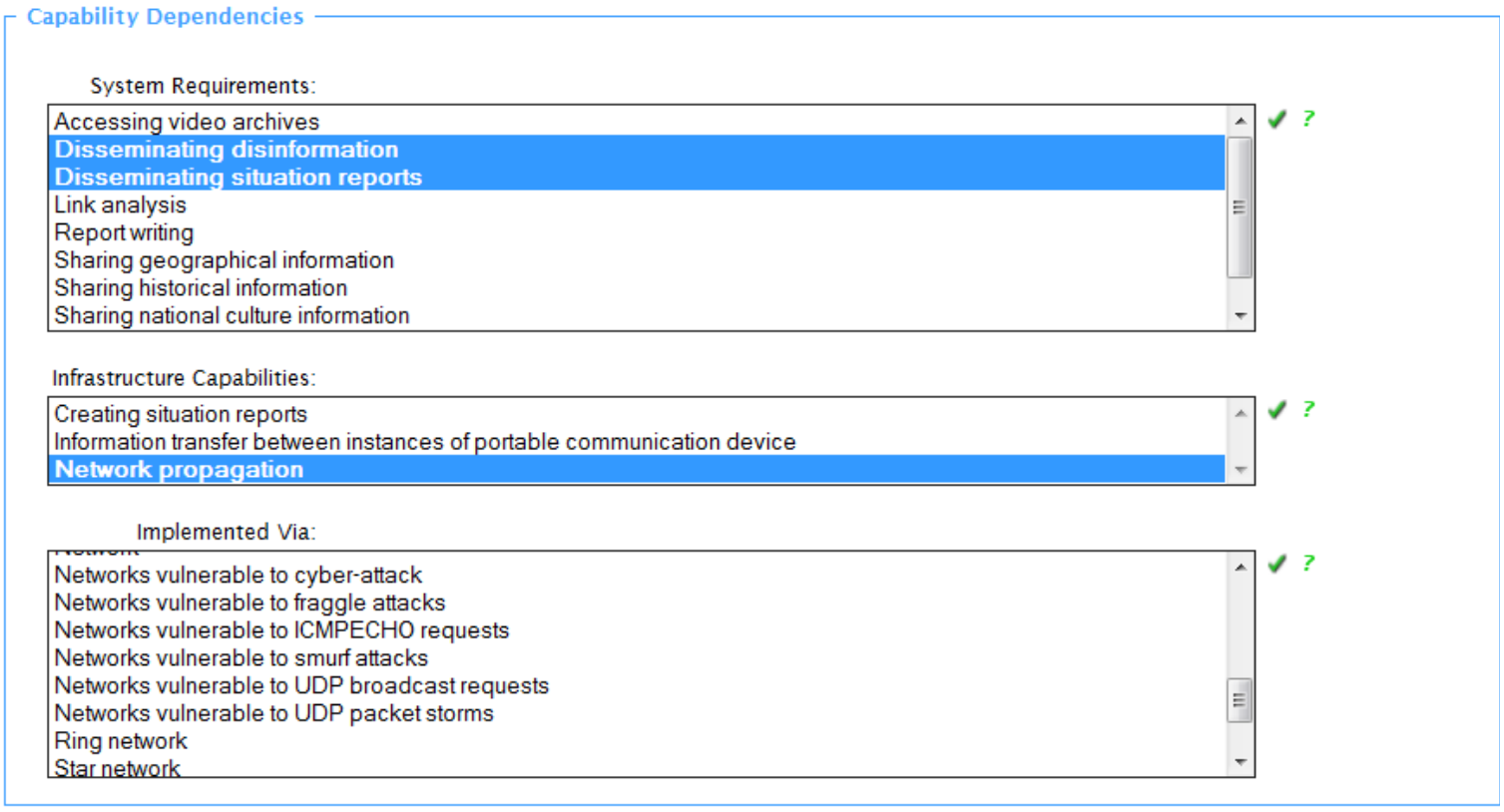


Figure 39. *Selecting 'Disseminating disinformation' from the 'System Requirements' menu in addition to 'Disseminating situation reports' results in additional types of vulnerable network appearing in the'Implemented Via' menu, so long as 'Network propagation' is selected.*

When a means of implementation is selected from the 'Implemented Via' menu, the formal system model is extended to include, not only the fact that an instrumentality of this type is part of the artifact infrastructure of the system being described, but the fact that such an instrumentality supports the infrastructure capability and system requirement that have been identified in association with it[22]. From the knowledge that this support relation exists we can infer counterfactual or hypothetical results about the behavior of the system in question. For example suppose that the user selects 'disseminating situation reports' => 'network propagation' => 'bus network':

[22] We are able to do this even in cases where multiple selections are made for each menu because the Knowledge Base keeps track of the logical dependencies between menu infills. For instance, because the logical dependency that enabled SIDECAR to populate the 'Implemented Via' with ordinary computer networks based on a user selection of 'Network Propagation' is encoded in the Knowledge Base as the knowledge that computer networks are needed for network propagation, and because we preserve a record of the fact that 'network propagation' is currently selected in the 'Infrastructure Capabilities' menu, we can recover 'Network Propagation' as the selected infrastructure capability supported by any network type selected from the 'Implemented Via' menu.

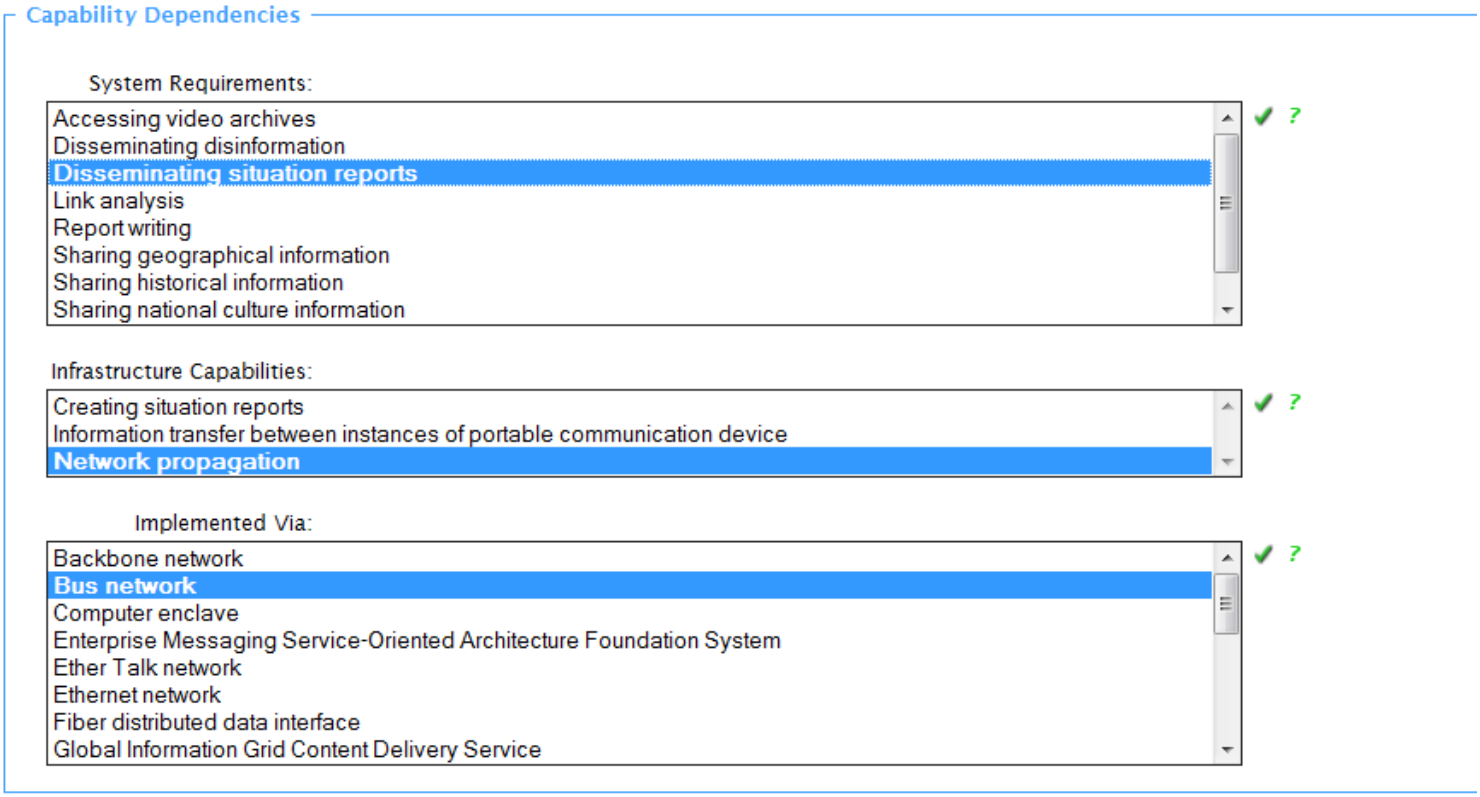


Figure 40. *Selecting 'Bus network' results not only in SIDECAR recording that such a network is part of the DCGS-MC, but that it supports network propagation, and the system performance requirement of disseminating situation reports.*

SIDECAR records the fact that the DCGS-MC subsumes a bus network which serves the operational function of network propagation within the system, and which meets the system performance requirement of disseminating situation reports. Among other inferences, this will support the conclusion that a report is disseminated via computer network in the DCGS-MC, a bus network will be the network type used for this purpose.

4.12 Architecting Option Visualization

The functionality that has been describe so far in relation to the capability dependency section is primarily descriptive, in that it assumes that the user is describing a more-or-less fully architected system. There is no reason, however, that the range of options inferred by SIDECAR in the context of known requirements could not be aggressively leveraged for actual design. If the user wants to compare options with respect to a known set of features, it is fairly easy to generate a query that does this, and to present the results in tabular form. As a near-term implementation, we've provided a feature that enables a user to select multiple elements from a capability discussion menu and run a saved 'profiling query' that returns indicated features for these elements. For example, if, having selected 'Disseminating situation reports', the user goes on to select 'Information transfer between instances of portable communication device', among the candidate means of implementation appearing in the 'Implemented Via' menu are numerous

communication standards. Here, the user has selected the four extant IEEE 802.11 communication standards: a, b, g, and n.

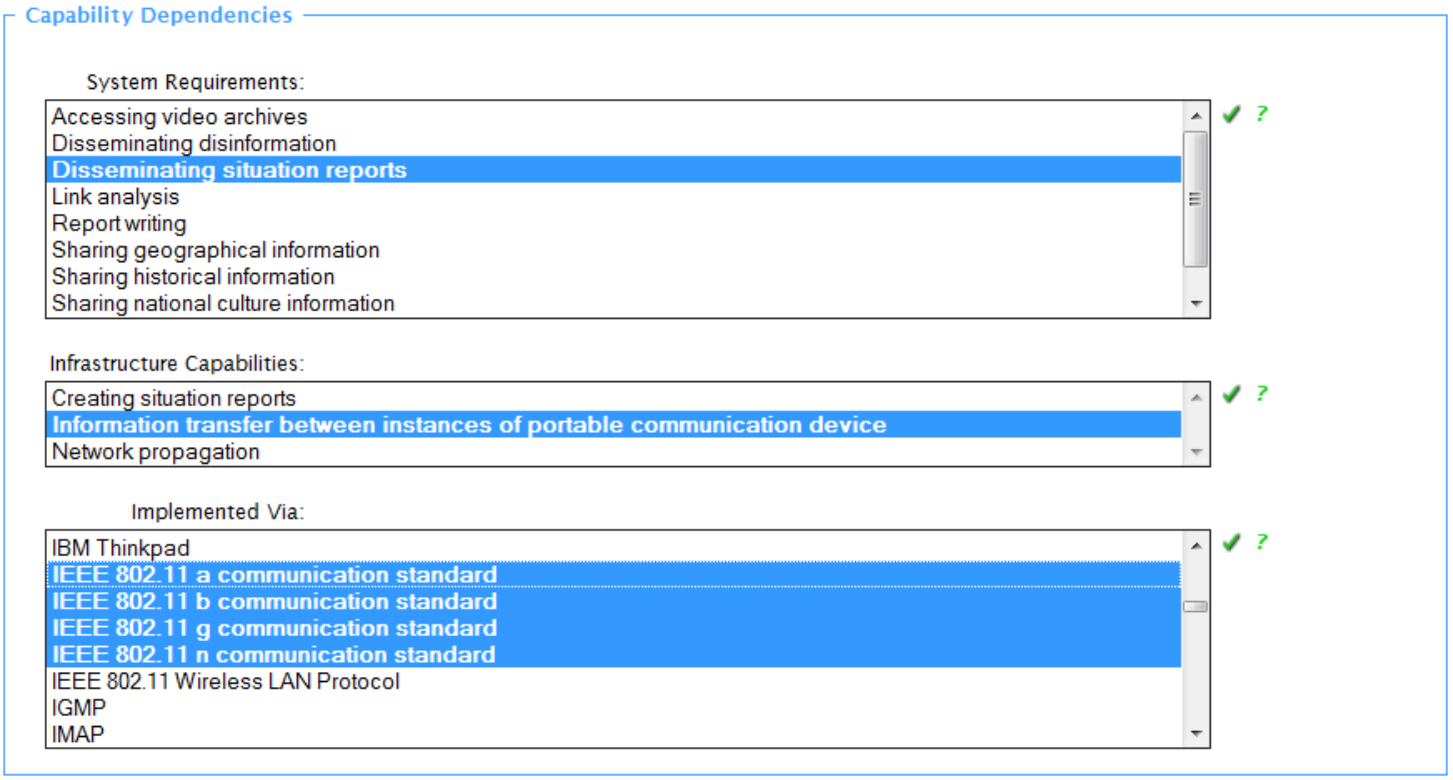
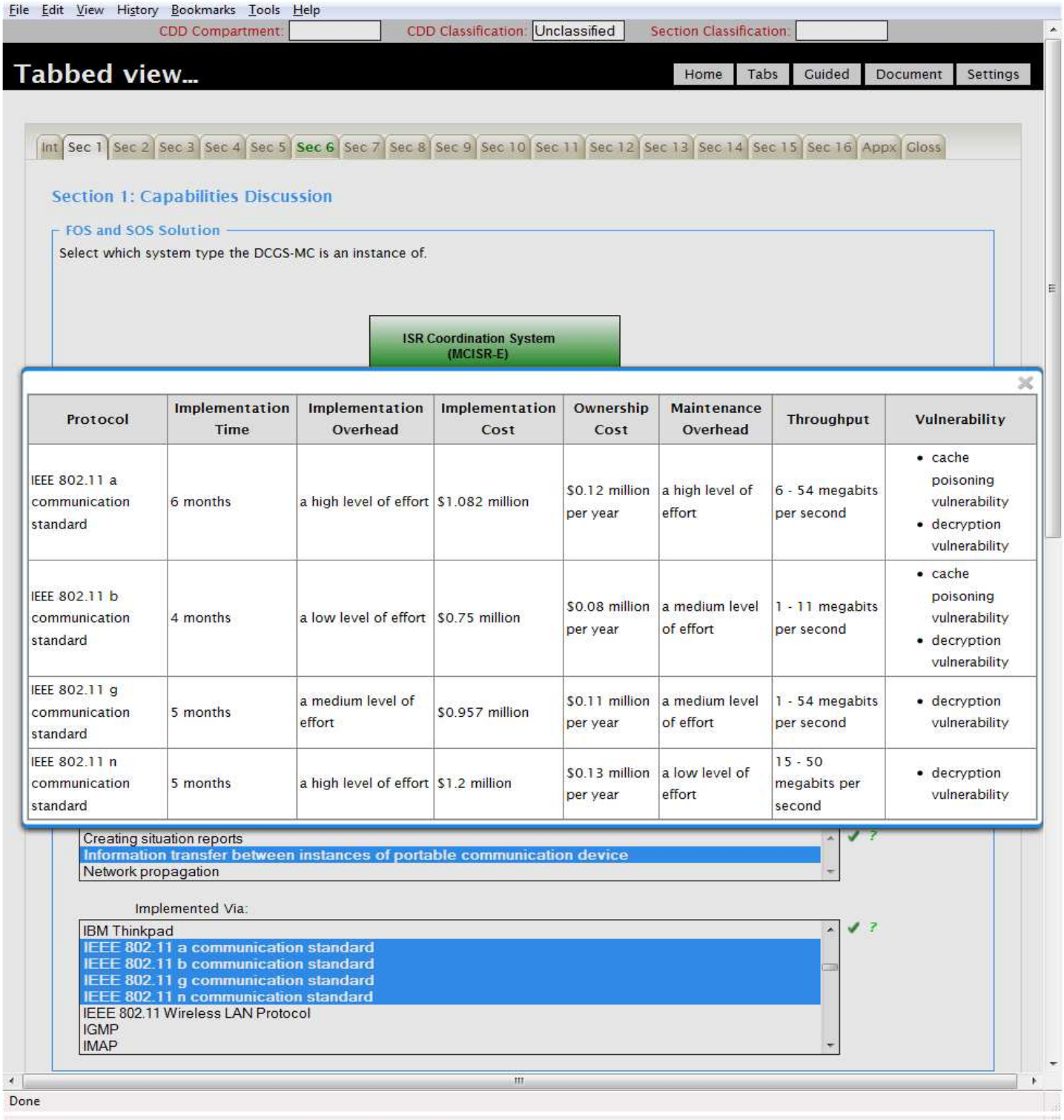


Figure 41. *Among the items that populate the 'Implemented Via' menu as the result of selecting 'Information transfer between instances of portable communication device' from the 'Infrastructure Capabilities' menu are the four IEEE 802.11 communication standards: a, b, g, and n.*

By clicking the green question-mark '**?**' icon that appears to the right of the menu, the user is able to run a saved query that polls the implementation time, implementation overhead, implementation cost, ownership cost, maintenance effort, throughput, and vulnerability for each standard, and returns the result in tabular form (see Figure 42).



Protocol	Implementation Time	Implementation Overhead	Implementation Cost	Ownership Cost	Maintenance Overhead	Throughput	Vulnerability
IEEE 802.11 a communication standard	6 months	a high level of effort	$1.082 million	$0.12 million per year	a high level of effort	6 - 54 megabits per second	• cache poisoning vulnerability • decryption vulnerability
IEEE 802.11 b communication standard	4 months	a low level of effort	$0.75 million	$0.08 million per year	a medium level of effort	1 - 11 megabits per second	• cache poisoning vulnerability • decryption vulnerability
IEEE 802.11 g communication standard	5 months	a medium level of effort	$0.957 million	$0.11 million per year	a medium level of effort	1 - 54 megabits per second	• decryption vulnerability
IEEE 802.11 n communication standard	5 months	a high level of effort	$1.2 million	$0.13 million per year	a low level of effort	15 - 50 megabits per second	• decryption vulnerability

Figure 42. *Selecting all four communication protocols and clicking the green question mark to the right of the menu results in a query being run that returns projected implementation time, implementation overhead, implementation cost, ownership cost, maintenance overhead, throughput, and vulnerability for each member of the set. Results are returned in tabular form.*

We can readily imagine extending this feature to allow active architecting by the user throughout the course of document development.

4.13 Operational Environment and Related Information

Below the Capability Discussion Subsection, the current CDD Knowledge Acquisition scheme provides for further knowledge capture concerning details of the operational environment of the proposed system.

In particular, we have subsection menus that allow the user to specify the operation type(s) supported, the expected operating environment type(s), the headquarters location, potential deployment locations, applicable supporting documents, and applicable Tier 1/Tier 2 Joint Capability Areas (JCAs). See Figure 43.

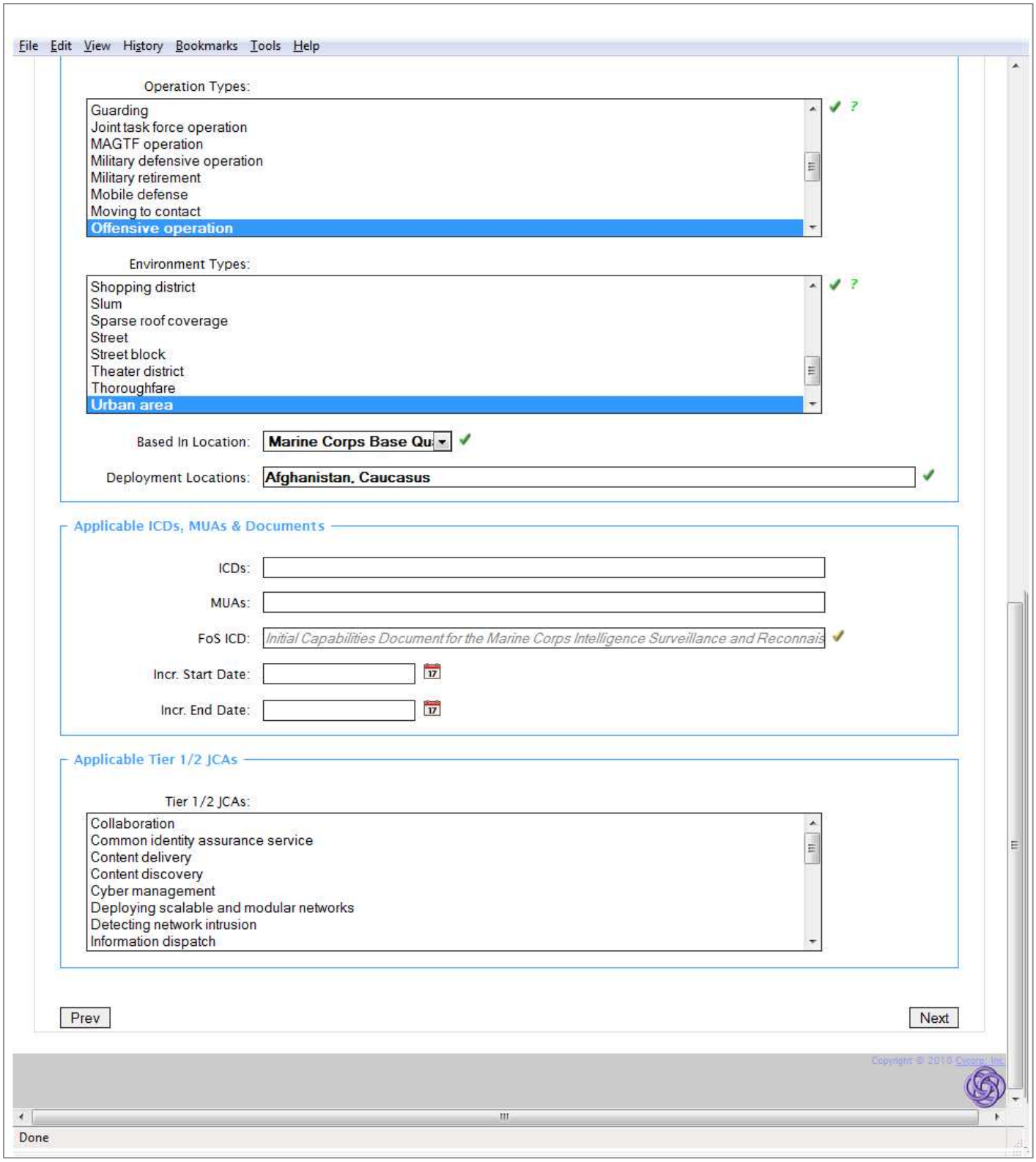


Figure 43. *The remaining subsections of the KA scheme for CDD section 1. The operation types, environment types, and Tier 1/2 JCAs all populate based on the choice from the FOS/SOS solution.*

All of the indicated menu populations are dynamic, and are generated on the basis of what SIDECAR is able to infer about the system.

For example, the Joint Capability Areas (JCAs) listed include the so-called 'net centric JCAs' [United States Department of Defense, January 2009, p. 37], on the grounds that these are particularly relevant to an Intelligence Analysis and Production System [United States Joint Chiefs of Staff, December 2008], which the DCGS-MC is now known to be. The headquarters location list is a drop-down menu which includes the principal Marine Corps headquarters locations (Quantico, Camp Lejeune, etc.), because the DCGS-MC is now known to have Marine Corps sponsorship by dint of its inclusion in MCISR-E.

Further dynamic inference is possible for all user selections and field in-fills. As an example, notice that in the screen capture shown above, the user has entered both 'Afghanistan' and 'Caucasus' as potential deployment regions. This results in a SIDECAR's concluding in the documentation context that the DCGS-MC includes elements that are deployable to these regions.

Cyc, in turn, knows that to be deployable to a region entails deployability to any known (or provable) sub-region of that region. Thus, for example, after telling SIDECAR that the DCGS-MC has units deployable to the Caucasus, SIDECAR can infer that the DCGS-MC has units that are deployable to, say Armenia, since it already knows that Armenia is *in* the Caucasus.

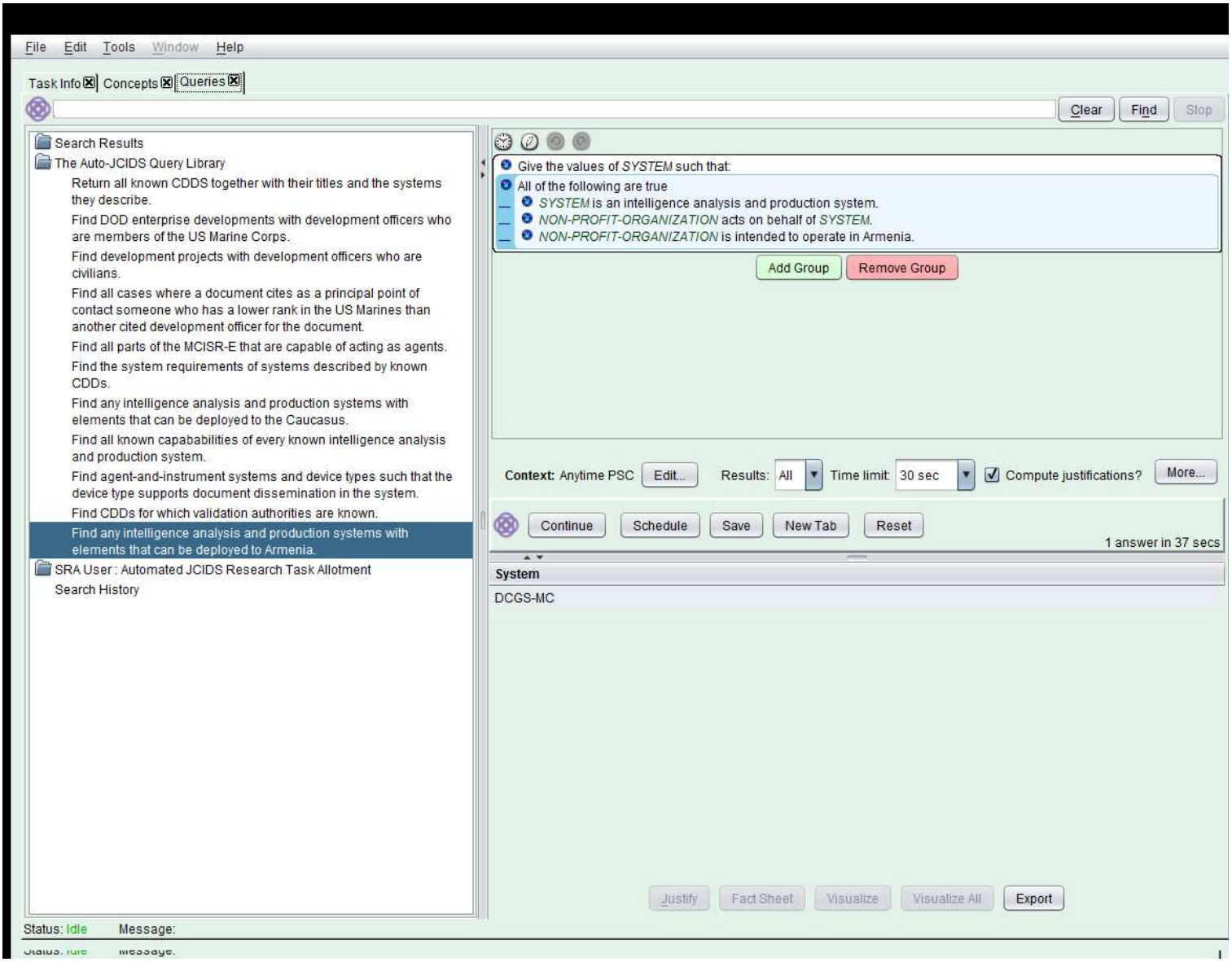


Figure 44. *Using the CAE to query for systems with units deployable to Armenia. The DCGS-MC is returned as an answer.*

Moreover, we can ask SIDECAR to explain the argument behind its reasoning – the set of reasoning steps justifying this conclusion. See Figure 45.

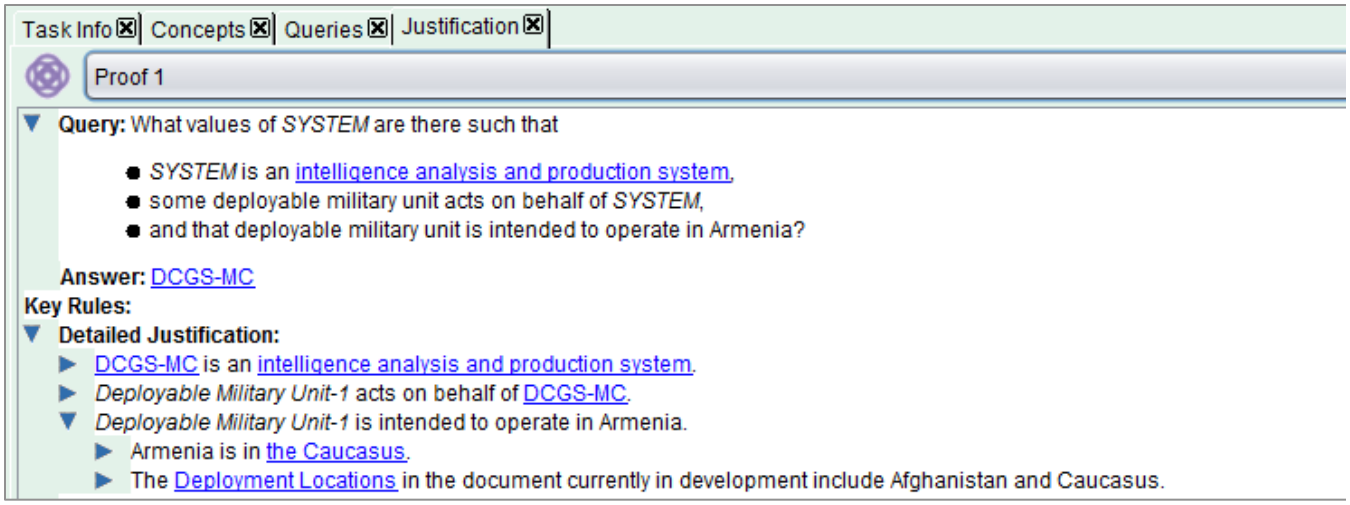


Figure 45. *The justification for the previous query. Armenia is part of the Caucasus, and Cyc knows that deployability to a region entails deployability to subregions of that region.*

4.14 Other Uses of Dependent Menus: Building KPP/KSA Tables

The SIDECAR Knowledge Acquisition vocabulary is sufficiently flexible to allow ontologists to implement more than one kind of population-dependency between different knowledge acquisition menus: a fact which makes it feasible to use menu dependencies for a variety of different purposes in JCIDS document development. For instance, one purpose for which they may be plausibly employed is in 'row-by-row' construction of the Key Performance Parameter tables and Key System Attribute tables in Section 6 of the Capability Development Document.

Here, in figure 46, is how SIDECAR gives the user the opportunity to select a KPP and then explain the parameter values and rationale for those values:

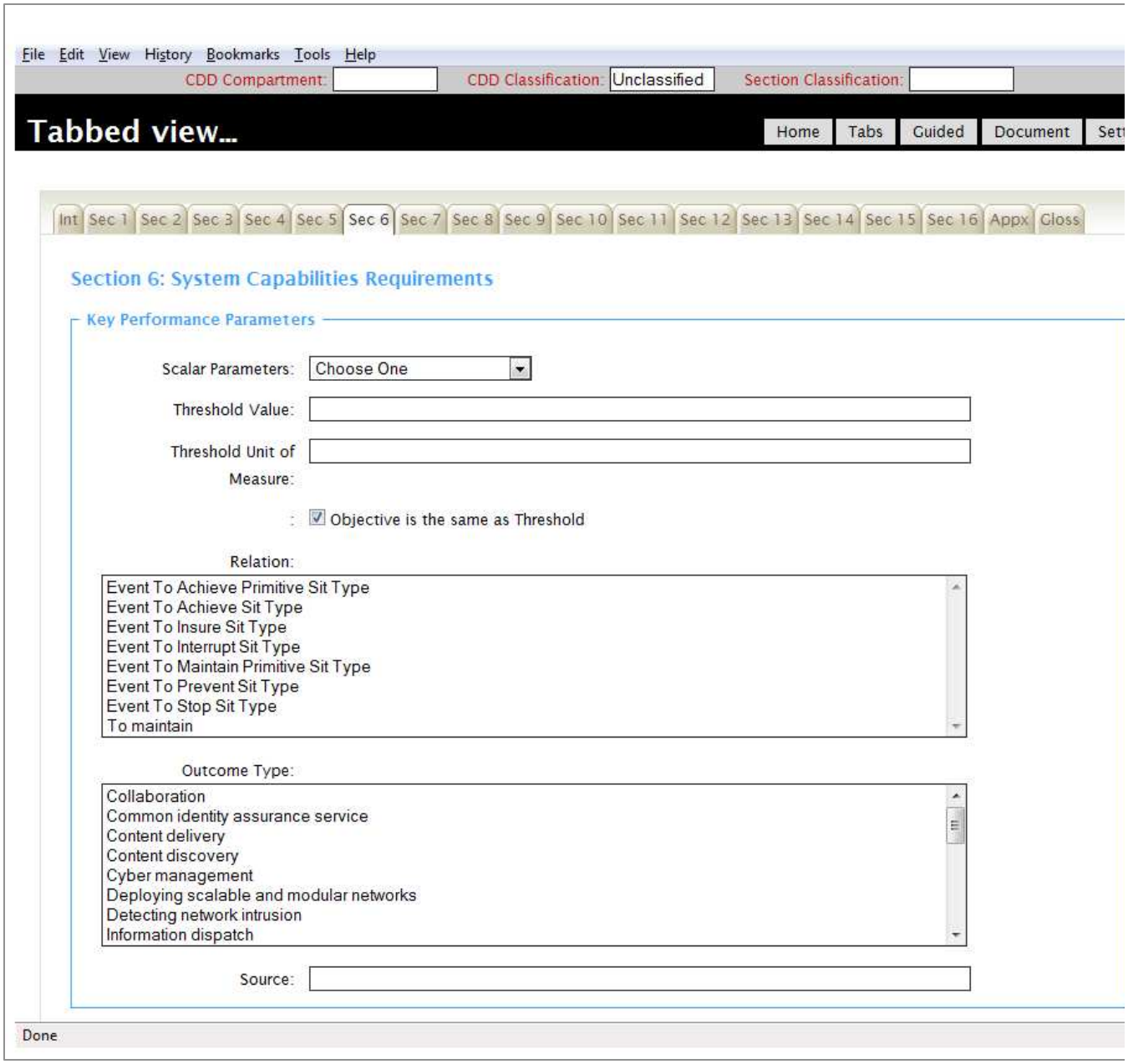


Figure 46. *The key performance parameters (KPP) subsection.*

The user begins by selecting a 'scalar parameter' – i.e., a salient system performance measures - from a menu of candidate metrics. The available choices are determined by the system type: for an intelligence analysis and production system, we currently generate a union of computer-related metrics and network throughput metrics. If the user selects one of these, say, 'CPU speed', then the threshold unit-of-measure field will automatically infill with a suitable unit-of-measure, such as TPS (transactions-per-second).

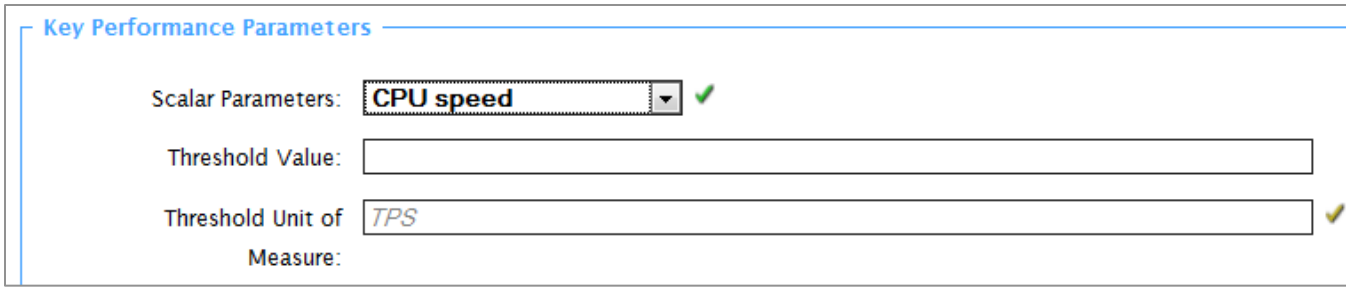


Figure 47. *As soon as a scalar parameter is selected, the threshold unit-of-measure field automatically infills with a suitable unit-of-measure.*

The user can then enter a numeric value for the metric.

Notice the check-box labeled 'objective is the same as threshold'. The default is for this to be checked: if it is unchecked, the assumption is that the system's objective value for the selected metric is not the same as the threshold value. Accordingly, a new set of KA objects appears that enables the user to input a different objective value:

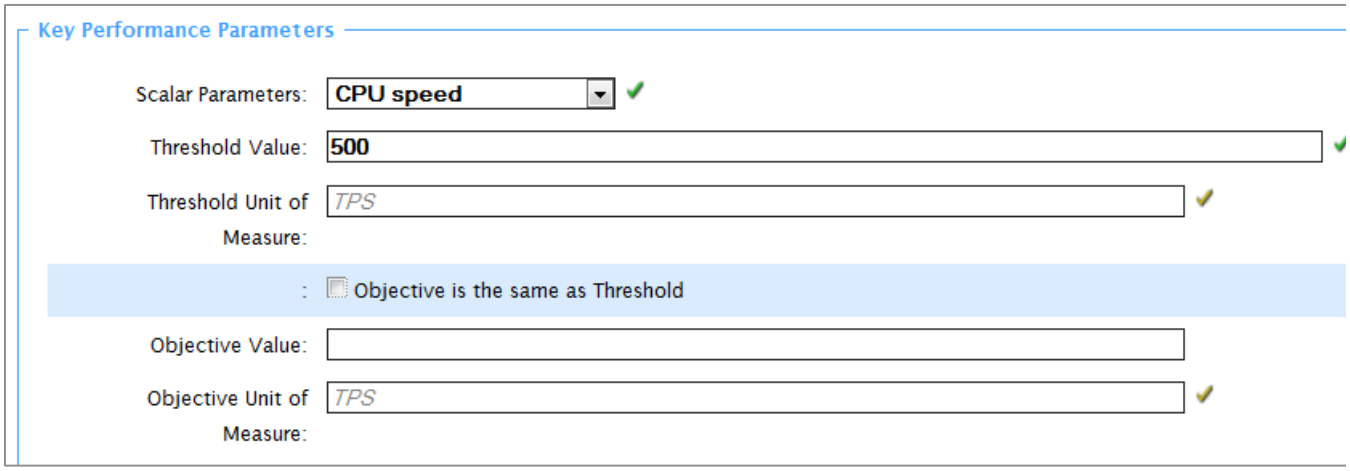


Figure 48. *Un-checking 'Objective is the same as Threshold' results in the objective fields becoming visible.*

This is an example of how *entire questions* appear or disappear depending on inferred numbers of answers to them. If the Objective is the same as the Threshold, asking the user to enter two different values is not a meaningful thing to do.

Finally, the user moves to the Rationale menus, to explain to SIDECAR *why* that parameter is constrained to have those particular values (vs. higher/lower/other ones).

Relation:

Event To Achieve Primitive Sit Type
Event To Achieve Sit Type
Event To Insure Sit Type
Event To Interrupt Sit Type
Event To Maintain Primitive Sit Type
Event To Prevent Sit Type
Event To Stop Sit Type
To maintain

Outcome Type:

Collaboration
Common identity assurance service
Content delivery
Content discovery
Cyber management
Deploying scalable and modular networks
Detecting network intrusion
Information dispatch

Figure 49. *The Rationale menus.*

The first of these is a set of purpose specifying relations – to maintain, to prevent, to insure, etc.[23]

The second menu gives a list of situation types which are, in fact, salient Joint Capability Areas that we identify on the basis of known features of the system being described (for intelligence analysis and production systems, the range of choices comprises the 'net-centric' JCAs, as it did in the Tier1/Tier2 JCAs menu).

In general, then, the user selects both a purpose predicate and an outcome type, for each KPP parameter that is being constrained.

[23] The family of relations used relate an event to a situation type. The event in the argument 1 position in this context is the *implementation* of the system being described.

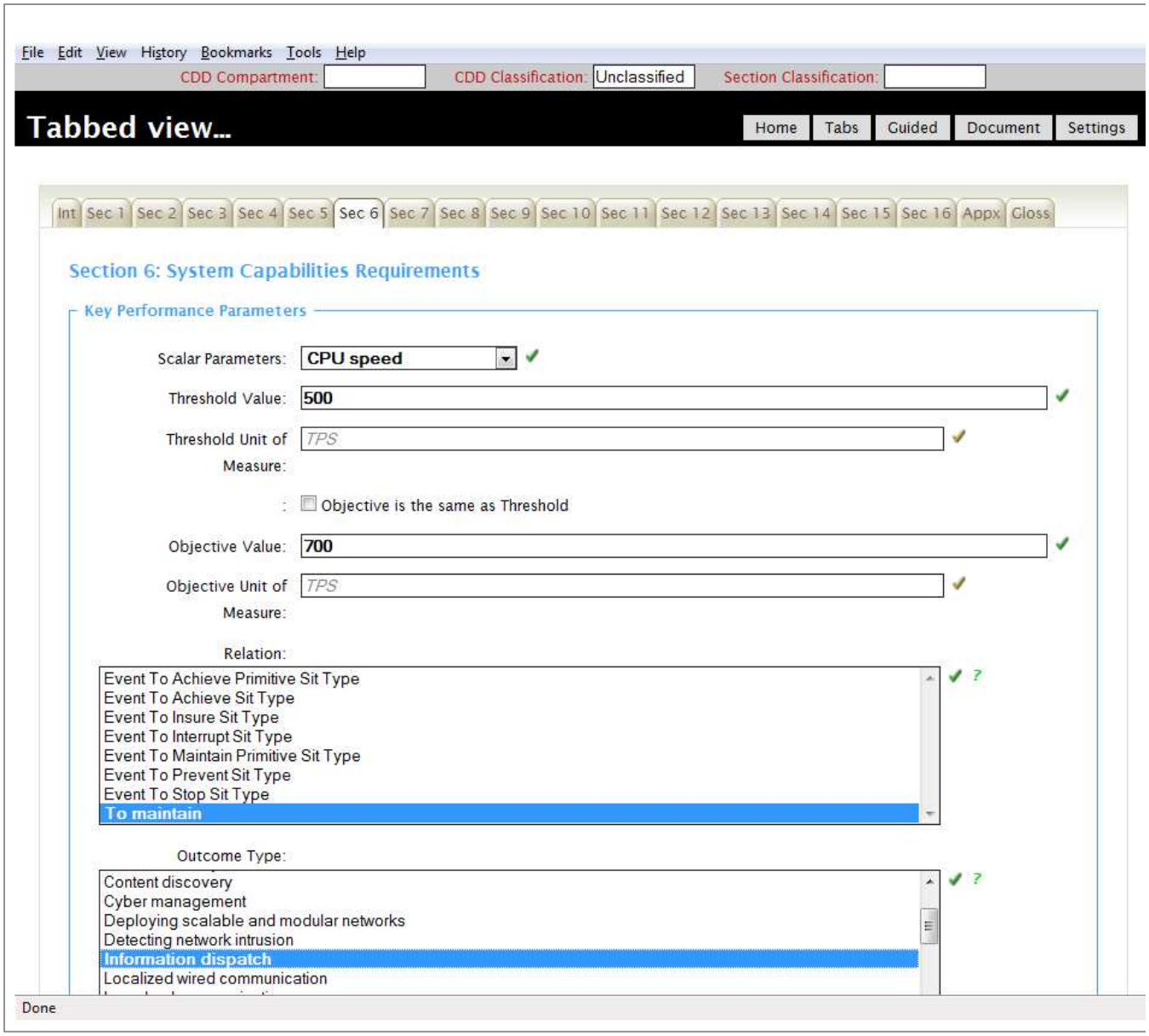


Figure 50. *Selecting a scalar parameter with threshold and objective values, together with exaclty one relation and exactly one outcome type defines a row in the KPP table.*

The joint selection of a unit-of-measure, threshold and (if different) objective values, a purpose predicate, and a situation type lets SIDECAR infer a couple of things. First, the choice of a threshold value relative to a metric tells SIDECAR that all components within the proposed system that are of the type suitable for the metric have *at least* the value on that metric that is specified as the threshold value. Thus, e.g., since 'CPU speed' applies to computers, entering a threshold value for CPU speed tells the Cyc system that any computer that is included in the DCGS-MC will have a CPU speed of at least 500 TPS. Similarly, the choice of objective value relative to the same metric tells SIDECAR that

all components within the system that are of the type suitable for the metric have precisely the objective value selected – e.g., that an entry of '700' in this context means that any computer in the DCGS-MC has a CPU speed of exactly 700 TPS.[24] Finally the rationale selection tells the system that the *reason* the threshold and objective values are set as they are is so that the implementation of the proposed system will have the indicated effect (maintain, prevent, etc.) on the cited situation type. Thus, if the user selects 'to maintain' and 'information dispatch', the Cyc system will know that the reason that the CPU speed of any computer in the DCGS has a minimal operating speed of 500 TPS and an ideal operating speed of 700 TPS is to maintain information dispatch.

This information can readily be compiled into tabular format in the document view. So far, the document has been a series of automatically synthesized English sentences. But SIDECAR can reason about the *modality* of the data it is placing in that document, and decide that some of it fits better as an Excel spreadsheet, or a node and link diagram, or – as in this case – a simple non-spreadsheet table. So SIDECAR fills out and then formats such a table (rather than a series of English sentences) and then inserts that table in the appropriate place in the English version of the CDD document, as shown in Figure 51. The table has columns labeled Rationale, Threshold, and Objective; each KPP parameter will have its own row in that table.

[24] The interpretive lifting rules are set up so that Cyc will conclude that any component of the relevant type has exactly the threshold value on the metric cited, *unless* an objective value is explicitly entered. The main utility of preserving the distinction derives from the fact that the documentation context in which the system description is entered is fundamentally *deontic*: that is, it contains a description of the way the system *should* be working under ideal operational conditions. At some later point in the modeling enterprise (perhaps at one of the incremental milestones), it would be plausible to compare an *actual* description of the system with this ideal one. Reasonable questions to ask under those circumstances would be (i) whether the system is performing in accordance with minimal (threshold) specifications and (ii) whether the system is performing according to optimal (objective) specifications. Such context-comparing questions could be authored and stored in the CAE.

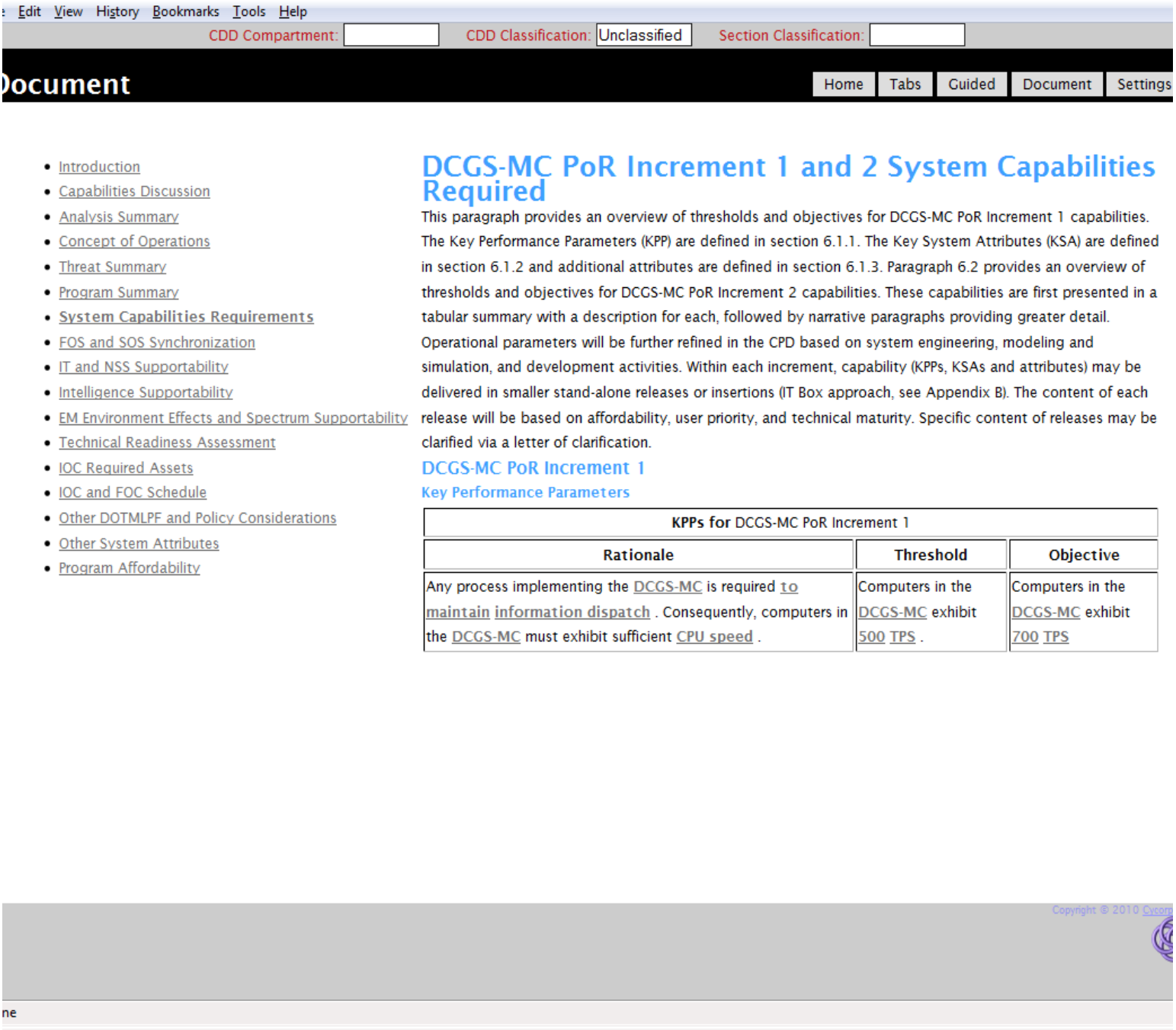


Figure 51. *The first row in the KPP table, as it appears in the synthesized English document version of the CDD.*

It should be noted in this context that columns of this menu are, for any given KPP, permanently bound together[25] into a single row in a table – thereby creating a

[25] In point of fact, for the KPPs (key performance parameters) it should not be *possible* to deselect or change a singleton menu choice from the scalar parameter menu (e.g., to change 'CPU speed' to 'Disk Capacity') while leaving the threshold and objective value field entries and rationale selections untouched.

requirement that there be and effective method for 're-setting' the menus and fields when the user is ready to move to the next row in the table. This convention will not be hard to implement: e.g., it is feasible to have both the KPP subsection and the KSA subsection equipped with 'move to next row' buttons where the effect of hitting the button would be to re-set all of the KA objects so that field entries and active selection highlights would disappear. In the current context, this would not mean that the user's entries had been overwritten or that his or her selections had been deselected, merely that the system had reset itself to receive the next input, with the user's original set of selections and entries being written to a table row (as well as being preserved in CycL assertions). To the extent that this has the potential for creating any confusion for the user regarding what has/hasn't been deselected, the right thing to do is probably to have a table view automatically open for the KPP and KSA subsections, so that the user will always be able to keep track of what is in the table at any given time. Such an interface features, like many of the features envisioned for the SIDECAR UI, is probably best negotiated with the prospective consumer – in this case, the CDD authors who will actually be using SIDECAR.

The Key System Attributes (KSA) subsection scheme is very similar to the KPP scheme, save that the scalar metric selection is replaced with a drop-down menu featuring applicable attribute relations such as 'performed by', 'assisting agent', 'mediators', and the like, and the threshold and objective value fields are replaced with drop-down menus of event types. Choosing a relation and an event type generates an assertion regarding capability: in the case of the threshold selection, what is inferred is that the system is (ordinarily) capable of playing the selected role in events of the selected type; in the case of the of the objective selection, we infer the capability plus the fact that this is the intended or target capability of the system; and we infer target capability for the threshold selection, too, unless the objective selection is different. The rationale is handled the same way as before.

It really should be an all-or-nothing affair: if a change is to be made, it has to be for all of the menu entries for a given row (granted, with sufficient brainstorming we can probably come up with a few outlier examples for which, say, certain threshold and objective values might be consistent with two or more choices of parameter, but they're going to be few and far between and by no means override this consideration). Therefore: if the user makes a change in the scalar parameter menu, they should be compelled to edit everything else. For the KSAs, where the initial choice involves an actorslot and the threshold and objective values are activity types that are selected from menus, we have a little more leeway: though we should flag the user if a change of role ("actor-slot") selection renders what had been meaningful threshold and/or objective value selections nonsensical.

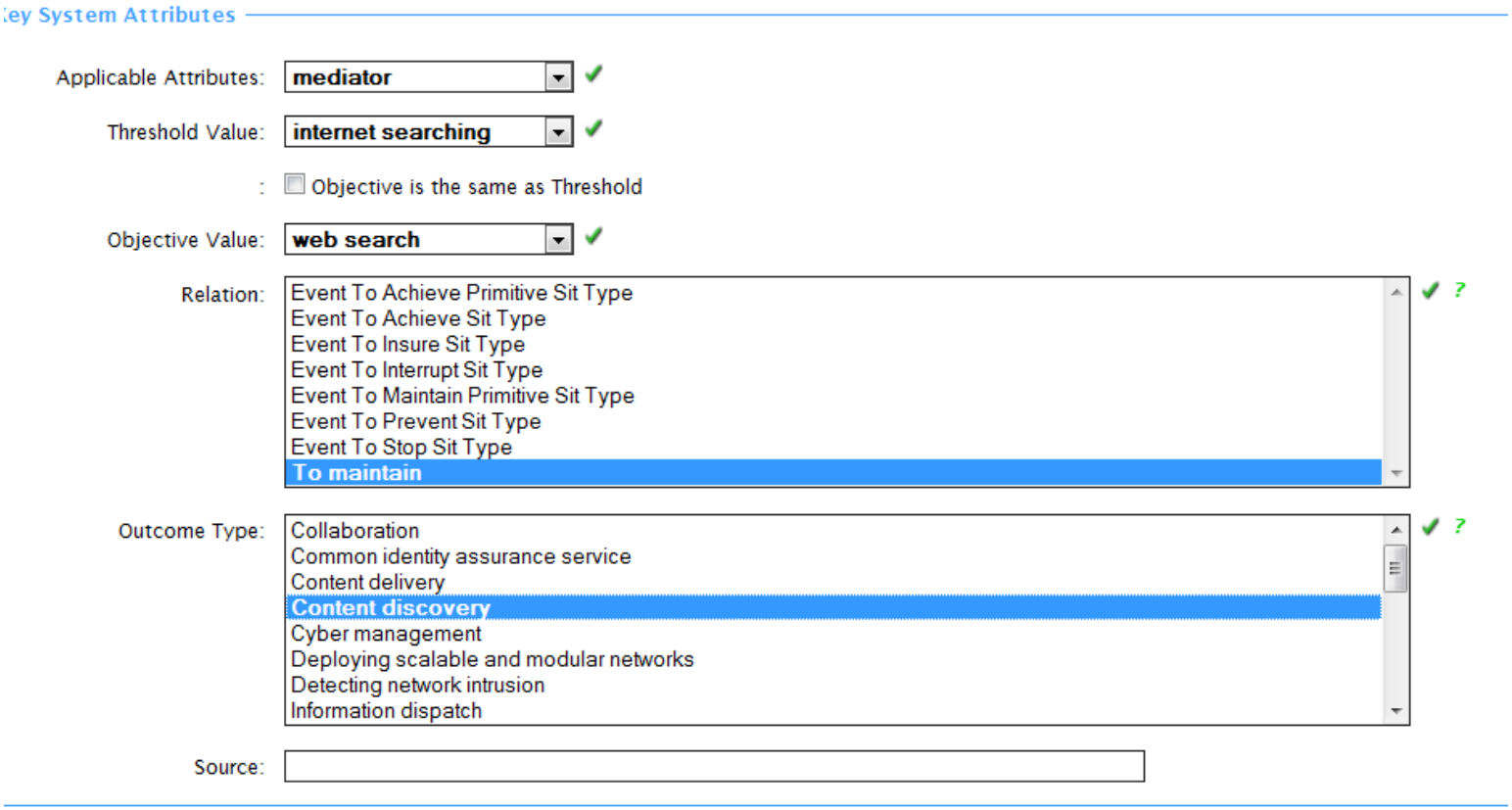
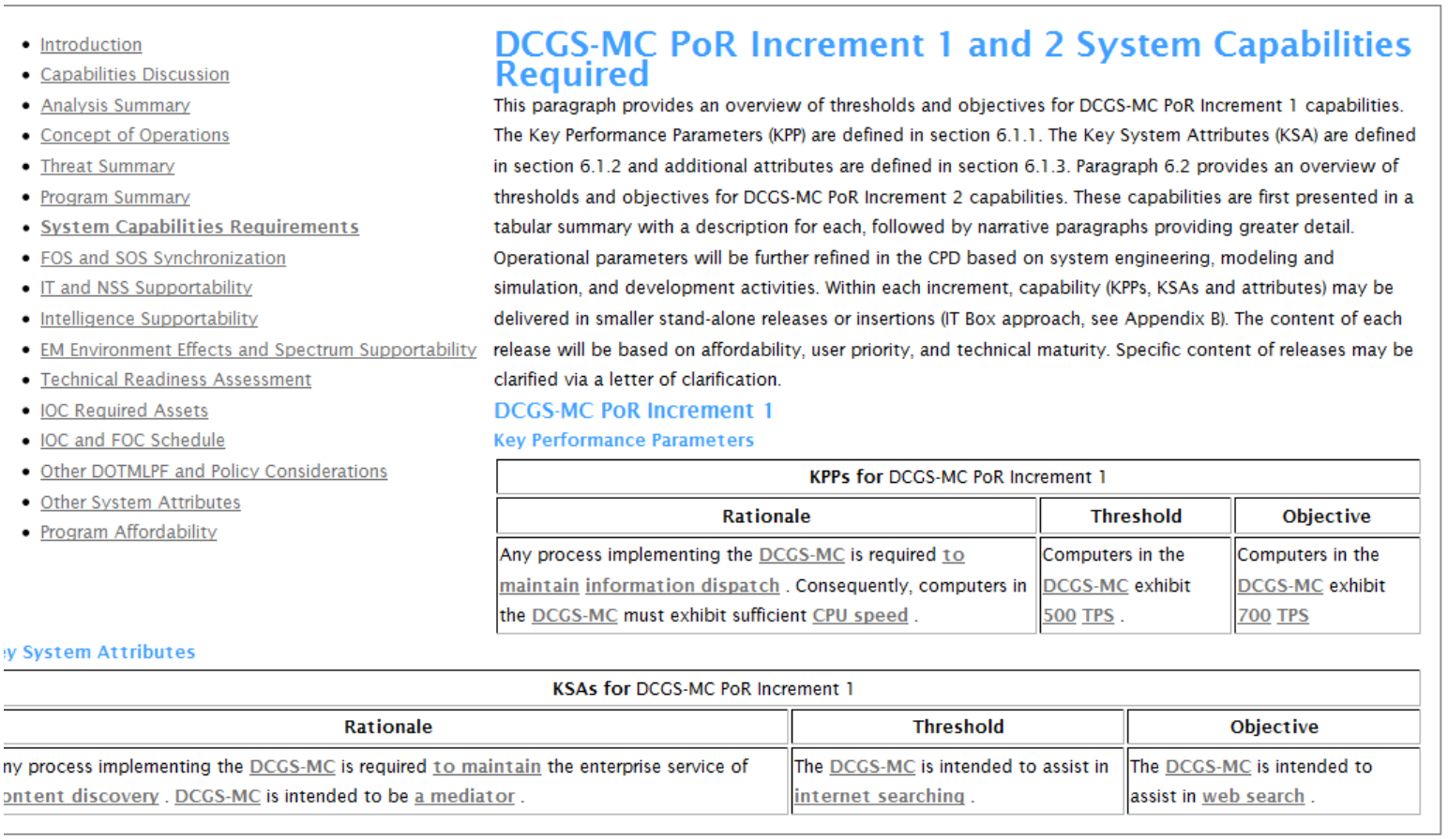


Figure 52. *Key system attribute (KSA) selections.*

Again, it is easy to compile this information into tabular form in the document view.



- Introduction
- Capabilities Discussion
- Analysis Summary
- Concept of Operations
- Threat Summary
- Program Summary
- **System Capabilities Requirements**
- FOS and SOS Synchronization
- IT and NSS Supportability
- Intelligence Supportability
- EM Environment Effects and Spectrum Supportability
- Technical Readiness Assessment
- IOC Required Assets
- IOC and FOC Schedule
- Other DOTMLPF and Policy Considerations
- Other System Attributes
- Program Affordability

DCGS-MC PoR Increment 1 and 2 System Capabilities Required

This paragraph provides an overview of thresholds and objectives for DCGS-MC PoR Increment 1 capabilities. The Key Performance Parameters (KPP) are defined in section 6.1.1. The Key System Attributes (KSA) are defined in section 6.1.2 and additional attributes are defined in section 6.1.3. Paragraph 6.2 provides an overview of thresholds and objectives for DCGS-MC PoR Increment 2 capabilities. These capabilities are first presented in a tabular summary with a description for each, followed by narrative paragraphs providing greater detail. Operational parameters will be further refined in the CPD based on system engineering, modeling and simulation, and development activities. Within each increment, capability (KPPs, KSAs and attributes) may be delivered in smaller stand-alone releases or insertions (IT Box approach, see Appendix B). The content of each release will be based on affordability, user priority, and technical maturity. Specific content of releases may be clarified via a letter of clarification.

DCGS-MC PoR Increment 1

Key Performance Parameters

KPPs for DCGS-MC PoR Increment 1		
Rationale	**Threshold**	**Objective**
Any process implementing the DCGS-MC is required to maintain information dispatch . Consequently, computers in the DCGS-MC must exhibit sufficient CPU speed .	Computers in the DCGS-MC exhibit 500 TPS .	Computers in the DCGS-MC exhibit 700 TPS

y System Attributes

KSAs for DCGS-MC PoR Increment 1		
Rationale	**Threshold**	**Objective**
ny process implementing the DCGS-MC is required to maintain the enterprise service of ontent discovery . DCGS-MC is intended to be a mediator .	The DCGS-MC is intended to assist in internet searching .	The DCGS-MC is intended to assist in web search .

Figure 53. *The document view, showing the first row in the KPP table and the first row in the KSA table.*

4.15 The Glossary and Its Uses

The final tab in the CDD KA scheme points to a glossary. Here, every term that is referenced in the CDD is tracked, together with its documentary explanation (where available from the Knowledge Base) in alphabetical order. For example, by clicking on the 'J' tab in the glossary, the user sees a list of terms including the acronym 'JROC' which was introduced in the introduction section.

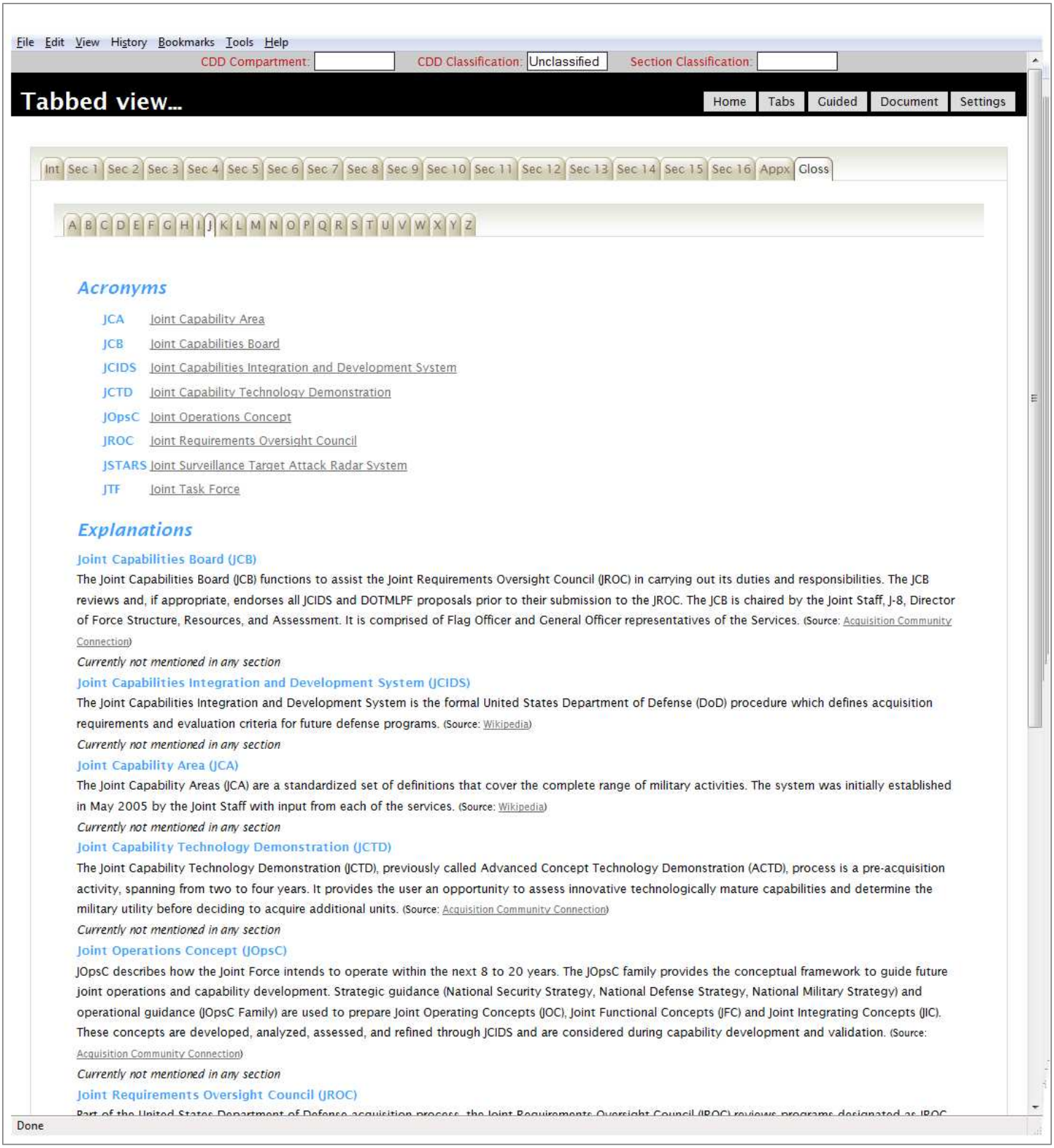


Figure 54. *The tab of the glossary for terms beginning with the letter "J".*

Clicking on the full name ('Joint Requirements Oversight Council') that appears alongside the acronym takes one to the glossary explanation for the term:

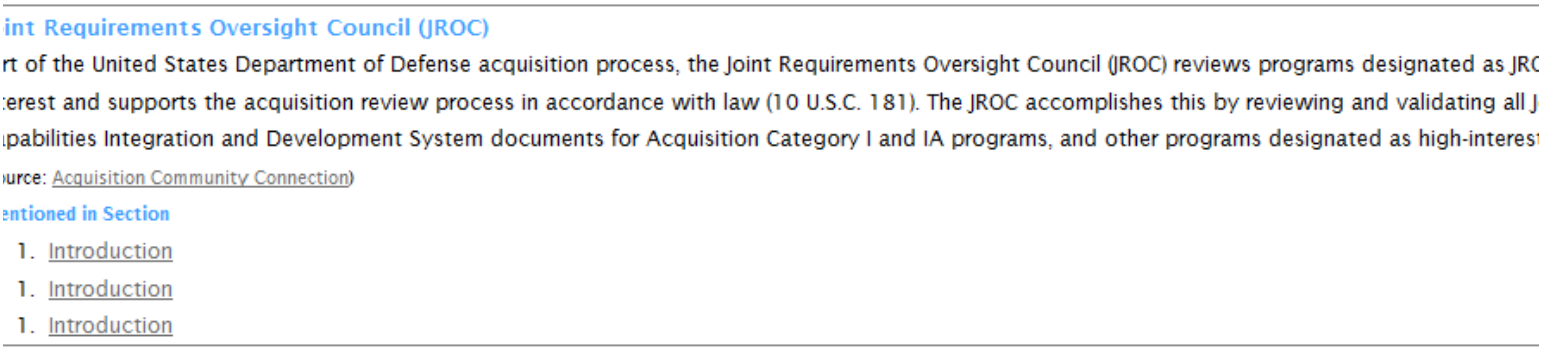


Figure 55. *JROC explanation.*

Note that 3 hotlinks appear here, now, one for every occurrence of the term that is (as of that moment) known to appear in the document. Clicking one of those links returns the user to the KA object in the tabbed view that corresponds to the reference.

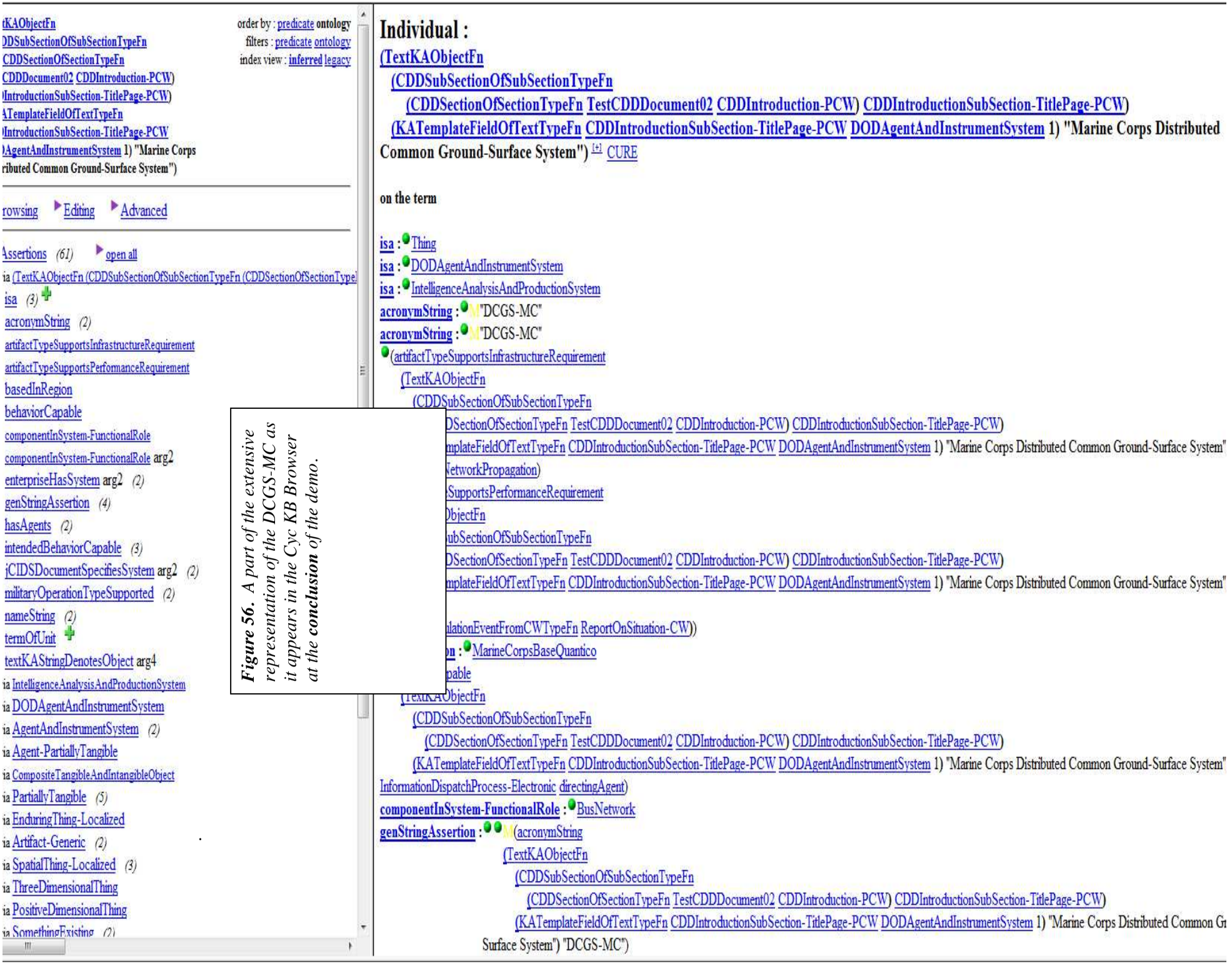


Figure 56. *A part of the extensive representation of the DCGS-MC as it appears in the Cyc KB Browser at the* ***conclusion*** *of the demo.*

5 Conclusion: Next Steps

5.1 Near-Future Extensions

As things presently stand at the conclusion of our initial (Phase 1) effort, a number of near-term follow-up steps are indicated for the SIDECAR system: we have divided these up into 6 categories: "next steps" regarding document representation, KA infrastructure, functional/capability model, user interface, document (and intermediate state) sharing/export, and policy/technology change management.

5.1.1 Next Steps in Scope/Depth of Document Representation:

One very small *conceptual* step to take would be to extend the current SIDECAR system so that it was able to generate an entire CDD front to back, instead of just some of the sections. This represents mostly a level-of-effort change from what we have done in Phase 1, and, though it is worthwhile to do, as we go through those additional sections the process is likely to lead to few if any new insights.

The current infrastructure that allows us to auto-generate CDD substructures and associated reasoning contents could be, and needs to be, extended so that it is also possible to generate the appropriate document substructures for other types of relevant JCIDS document types: ICDs, CPDs, MUAs, AoAs, etc. Besides treating each of these one by one, much additional leverage could be gained by modeling the various interdependencies between multiple types of JCIDS documents for interrelated programs and, of course, for the different evolutionary stages of the *same* program. This is more challenging than the task described in the previous paragraph, and more likely to lead to some important new lessons learned.

Something else which could be begun immediately – or deferred – is extension of the SIDECAR system to other Joint Forces and Services (Army, Air Force, Navy) documents/programs.

5.1.2 Next Steps in Knowledge Acquisition Infrastructure:

As the two 5.1.1 tasks are proceeding, there will inevitably be several new aspects to the knowledge acquisition paradigm that engender changes and expansions of our current SIDECAR KA system. Addressing these will require new code to be produced and new decisions about how various sorts of inputs can be done by the user (new gestures, icons, entry paradigms, display modalities, etc.) For example, in what ways, and to what extent, should the full range of Cyc's natural language understanding capabilities be harnessed? natural language generation capabilities? reasoning by analogy capabilities?

5.1.3 Next Steps in Functionality Model:

The more detailed aspects of the functionality model (e.g., the assumptions underpinning the knowledge capture from the menu dependencies in the Capabilities discussion section of a CDD) need to be reviewed with experienced JCIDS development officers and subject matter experts. We believe that very good/experienced JCIDS development officers will be much better at these tasks than average ones, who in turn will be much better than new ones; the goal of this series of next steps 5.1.3 is to make sure that SIDECAR helps capture as much of the expertise as possible, and is as much help to experienced JCIDS development officers as to new ones.

Analogous work with the appropriate SMEs will need to be undertaken for all of the remaining CDD subsections, and for the relevant requirements analysis components of all additional JCIDS document types (5.1.1) that are understaken to represent.

We expect a great deal of transfer learning to occur, both in our project staff and in SIDECAR, so that as 5.1.3 proceeds the rate at which the remainder of 5.1.3 proceeds will keep increasing.

5.1.4 Next Steps in User Interface:

The ultimate goal of 5.1.4 is to afford Cyc the intelligence needed to generate and 'drive' a user interface, based on knowledge of a KA scheme, a user model, and information about other contingent features of the KA environment. That is, Cyc (and the SIDECAR application in Cyc) would dynamically, as it runs, decide on where on the screen to place various things, what questions to include and when and in what modality, and so on.

Ideally, we would like to make the interface code general enough and robust enough to allow a Cyc ontologist with limited training in interface design to define a knowledge acquisition schema in the KB that can be immediately realized in a first-approximation interface rendering. Accomplishing this will require us to revisit some of the current assumptions of the interface code in addition to possibly modeling additional features of UI activity declaratively in the Cyc KB.

5.1.5 Next Steps in Document Export and Intermediate Representation Export

This task involves integrating SIDECAR's output with various document generation and viewing applications (e.g., Facchina Global's CDTM). It also involves tying in to Semantic Web and similar efforts, to maximize the sharing of the formal model's assertions, not just the final English document.

5.1.6 Next Steps in Policy/Technology Change Management

This involves more explicitly representing the policies, technologies, geopolitics, etc., which collectively define the complex terrain and environment in which the JCIDS documents live and evolve. Another powerful use of this level of modeling would be to handle various sorts of "what if?" hypothetical reasoning, exploring the ramifications of proposed policy changes, potential technology changes, and so on.

Focusing on *near term* extensions, this means ontology building and KB building which only trained ontologists and knowledge engineers would be able to do, or make use of, but which would enable vastly more powerful updates to the system (albeit involving those individuals), and which would lay the foundation for producing interfaces and tools by which subject matter experts (rather than ontologists and logicians) could make those changes (hence this aspect is expanded on below, in Section 5.2 Longer Term Extensions).

5.2 Longer-term Extensions

Some of the longer term extensions are directly related to making the system deployable:

- Once the prototype has been fleshed out to the point where it can be used to generate an entire CDD, the time will have come to begin integrating it with the relevant USMC databases, DOD databases, and other related (e.g., NGA) information sources.
- Another task appropriate to carry out, which will help greatly with deployability, will be to produce an easier to use *ad hoc* query answering interface – essentially an augmented and specialized version of the Cyc Analytic Environment (CAE) described above, tailored to make it less cumbersome for the intended end users.
- As the system begins to be deployed, it will become increasingly important to have a solid "regression testing" test suite, to detect problems that crop up as early as possible. That same sort of re-running of past JCIDS documents runs (through SIDECAR) is also more or less what will be required to produce dynamic alerts when some change in policy, technology, geopolitics, DISA IAVA/IAVB warnings, etc. makes some JCIDS documents no longer compliant and consistent, meaning that their respective programs may have become vulnerable or out of compliance.
- Documentation and training materials will need to be prepared, and, collaterally, the in-house expertise and capabilities to run and extend the technology should be developed. Cycorp's role should steadily devolve from custom application developer down to Tier 2 and eventually just Tier 3 support.

An additional task, which could be carried out in parallel with the foregoing, will be to augment the underlying Knowledge Base and the reasoning engine so that SIDECAR can support software architects who are tasked with *designing* new target systems, not just individuals tasked with documenting their designs. Some of this has happened already –

and is demonstrable in the current SIDECAR system – but much effort would be required to elevate this from the "interesting anecdote" level to anything approaching complete coverage which architects and designers could count on.

In the later stages of the development process, plausible enhancements include extending the SIDECAR system to support representation of policies and requirements modeling *by non-specialists* (i.e., lightly-trained personnel with limited experience in ontology development), and extending the aforementioned 'regression testing' feature to cover the re-running of documents with counterfactual models of the world reflecting *potential* changes in policies, requirements, priorities, new technologies, threats, and other "real world" events, in order to support analysis of the impact of the change. This extension of the 5.1.6 work would create JCIDS-wide accessibility to those powerful tools.

As an example, suppose that the current *suggestion* that NCES (net-centric enterprise services) systems be the preferred choices for messaging and content delivery is replaced – or being considered to be replaced – by a *requirement* that they be given priority. (See Figure 57.) Then those CDDs and other JCIDS documents which are out of compliance would be flagged as such, and appropriate alerts could be emailed to the proper contacts.

Report Dissemination ➔ Network Propagation
➔ ***suggest*** relevant NCES (net-centric enterprise service) systems whose primary function is messaging or content delivery

Change in policy to ***require*** all DOD intelligence analysis and production systems to use of at least one such NCES system.

Change in policy to ***require*** all DOD systems with an intelligence exploitation requirement to use ***at least two different*** such systems.

Figure 57. Changes to policies, requirements, technical capabilities, costs, etc. can be made declaratively to the KB. These can then signal alerts when new or pre-existing programs are out of compliance, and can also be run in hypothetical "what if" contexts to assess the potential impact of such changes.

Suppose that there is serious consideration, in some future year, due to increased hacking and intrusions and system compromises, to possibly requiring that each DOD system with an intelligence exploitation requirement use at least two different NCES systems redundantly. Then the set of current systems which would become out-of-compliance with such a change (including DCGS-MC) would be automatically identified, thus enabling the impact (and timing, cost, etc.) of that proposed policy change to be better assessed.

6 Bibliography

Enderton, H. B. *A Mathematical Introduction to Logic*, second edition. San Diego: Academic Press. 2001.

Gunderson, C. Let's Cure Defense Enterprise Information Technology Acquisition with a Dose of its Own Medicine. In *Proceedings of the AFCEA-GMU C4I Center Symposium*. (http://c4i.gmu.edu/events/reviews/2010/GMU-AFCEA-Agenda-2010.php) May 2010.

Hopkins, R. and Jenkins, K. *Eating the IT Elephant: Moving from Greenfield Development to Brownfield*. IBM Press. 2008.

Lenat, D. and Guha, R. *Building Large Knowledge-Based Systems: Representations and Inference in the Cyc Project*. Addison-Wesley. 1990.

Lenat, D., Witbrock, M., Baxter, D., Blackstone, E., Deaton, C., Schneider, D., Scott, J., and Shepard, B. Harnessing Cyc to Answer Clinical Researchers' *ad hoc* Queries. to appear in *AI Magazine,* 31, 3, Fall, 2010.

Masters, J. Structured Knowledge Source Integration and its application to information fusion. In *Proceedings of the Fifth International Conference on Information Fusion*, Annapolis, MD, July 2002.

Matuszek, C., Cabral, J., Witbrock, M., DeOliveira, J. An Introduction to the Syntax and Content of Cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, Stanford, CA, March 2006.

Panton K, Matuszek C, Lenat D, Schneider D, Witbrock M, Siegel N, and Shepard B. From Cyc to Intelligent Assistant, in (Cai Y and Abascal J, eds.), *Ambient Intelligence in Everyday Life*, Springer, 1-31, 2006.

Reed, S., Lenat, D.B. Mapping Ontologies into Cyc. In *AAAI 2002 Conference Workshop on Ontologies For The Semantic Web*, Edmonton, Canada, July 2002.

Robinson, J.A. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, **12**(1): 23-41. 1965.

United States Department of Defense. *Joint Capability Areas*. 12 January 2009.

United States Department of Defense. *Manual for the Operation of the Joint Capabilities Integration and Development System*. 31 July 2009.

United States Joint Chiefs of Staff. *Interoperability and Supportability of Information Technology and National Security Systems*. 15 December 2008.

United States Marine Corps. *Capability Development Document for Communications Emitter Sensing and Attacking System Increment 1*. 7 July 2006.

United States Marine Corps. *Capability Development Document for Distributed Common Ground/Surface System Marine Corps (DCGS-MC) Enterprise (Program of Record Increment 1)*. 15 January 2010.

United States Marine Corps. *Initial Capabilities Document for the Marine Corps Intelligences Surveillance Reconnaissance Enterprise (MCISR-E), Version 1.3*. 5 July 2007.

United States Navy. *Approved Solution Planning Directive (SPD) for Program Objective Memorandum (POM) -10 Warfighting Investment Program Evaluation Board*. 3 May 2007.

6.1 On-Line References

http://www.cyc.com/doc/handbook/oe/05-basic-guidelines-for-efficient-cycl.html
A section from the Cycorp Ontological Engineer's Handbook, providing basic guidelines and heuristics for authoring CycL

http://www.cyc.com/doc/handbook/oe/09-mistakes-to-avoid.html
A text section from the Ontological Engineer's Handbook, detailing some of the more common errors encountered in authoring CycL.

http://www.cyc.com/doc/tut/ppoint/WritingEfficientCycLPart1_files/v3_document.htm
Part 1 of a PowerPoint tutorial on authoring efficient CycL.

http://www.cyc.com/doc/tut/ppoint/WritingEfficientCycLPart2_files/v3_document.htm
Part 2 of the PowerPoint tutorial on authoring efficient CycL.

http://www.cyc.com/doc/tut/ppoint/ErrorsConstants_files/v3_document.htm
A Powerpoint tutorial covering some of the errors commonly encountered in representing CycL constants and treating variables.

http://www.cyc.com/doc/tut/ppoint/ErrorsSpecializationGeneralization_files/v3_document.htm
A Powerpoint tutorial covering some of the errors commonly encountered in rule authoring and asserting generalizations.

http://www.cyc.com/doc/tut/ppoint/OtherErrors_files/v3_document.htm
A Powerpoint tutorial covering other errors commonly encountered in authoring CycL.

http://www-formal.stanford.edu/jmc/elaboration/elaboration.html
From the papers of John McCarthy: an extended discussion of elaboration tolerance.

http://www.cyc.com/cyc/technology/whitepapers (see esp. 'The Cyc System: Notes on Architecture – Nick Siegel, Keith Goolsbey, Robert Kahlert, and Gavin Matthews)
This link can be used to download several useful whitepapers, including one concerning the Cyc System Architecture.

http://www.cyc.com/doc/tut/ppoint/InferenceLogicalAspects_files/v3_document.htm
A Powerpoint tutorial on the logical aspects of Cyc inference.

http://www.cyc.com/doc/tut/ppoint/InferenceIncompleteness_files/v3_document.htm
A Powerpoint tutorial on incompleteness handling in Cyc inference.

http://www.cyc.com/doc/tut/ppoint/InferenceResourceBounds_files/v3_document.htm
A Powerpoint tutorial on resource bounding in Cyc inference.

http://www.cyc.com/doc/tut/ppoint/InferenceHeuristics_files/v3_document.htm
A Powerpoint tutorial on optimizing efficiency in reasoning through inference heuristics.

http://www.cyc.com/doc/tut/ppoint/InferenceFeatures_files/v3_document.htm
A Powerpoint tutorial describing the role of microtheories in inference, and explaining the difference between forward and backward inference in Cyc.

http://www.cyc.com/doc/handbook/oe/14-using-cycl-queries.html
A section from the online Ontological Engineer's Handbook, concerning query inference parameters and inference tree interpretation, with additional tips for using CycL queries.

http://www.cyc.com/cycdoc/ref/inferencing.html
An online OpenCyc reference document concerning Cyc inference and the role of heuristic modules.

http://en.wikipedia.org/wiki/CycL
The CycL representation language underlying the Cyc knowledge base.

Appendix A: Acronym Glossary

- ACAT: Acquisition Category
- AI: Artificial Intelligence
- AoA: Analysis of Alternatives
- APB: Acquisition Program Baseline
- C2: Command and Control
- C3I: Command, Control, Communications and Intelligence
- C4I: Command, Control, Communications, Computing and Intelligence
- CAE: Cyc Analytic Environment
- CBA: Capabilities-Based Assessment
- CDD: Capabilities Development Document
- CDTM: Capability Development Tracking and Management
- CGS: Common Ground Station
- CONOPS: Concept of Operations
- CPD: Capability Production Document
- CycL: The Cyc representation language
- DCGS: Distributed Common Ground/Surface System
- DCGS-MC: Marine Corps Distributed Common Ground/Surface System
- DIB: DCGS Integration Backbone
- DOD: Department of Defense
- DODAF: DOD Architecture Framework
- DODISR-E: DOD Intelligence, Surveillance, and Reconnaissance Enterprise
- DOTMLPF: joint doctrine, organization, training, materiel, leadership and education, personnel, and facilities
- EMD: Engineering, management, and development
- FOC: Full Operational Capability
- FoS: Family of Systems
- GIG: Global Information Grid
- HER: Hazards of Electromagnetic Radiation
- HERF: Hazards of Electromagnetic Radiation to Fuels
- HERO: Hazards of Electromagnetic Radiation to Ordnance
- HERP: Hazards of Electromagnetic Radiation to Personnel
- IC: Intelligence Community
- ICD: Initial Capabilities Document, also Intelligence Community Directive
- IM: Insensitive Munitions
- IOC: Initial Operational Capability
- ISR: Intelligence, Surveillance, and Reconnaissance
- JCA: Joint Capability Area
- JCB: Joint Capabilities Board
- JCTD: Joint Capability Technology Demonstration
- JCIDS: Joint Capabilities Integration and Development System
- JOpsC: Joint Operations Concept(s)
- JROC: Joint Requirements Oversight Council
- JSTARS: Joint Surveillance Target Attack Radar System

- KA: Knowledge Acquisition
- KB: Knowledge Base
- KSA: Key System Attribute
- KPP: Key Performance Parameter
- MAGTF: Marine Air-Ground Task Force
- MCISR-E: Marine Corps Intelligence, Surveillance, and Reconnaissance Enterprise
- MDA: Milestone decision authority
- Mt: Microtheory
- MUA: Military Utility Assessment
- NART: Non-Atomic Reified Term
- NL: Natural Language
- NR-KPP: Network-Readiness Key Performance Parameter
- OOTW: Operations other than war
- ORD: Operational Requirements Document
- PISR: Persistent Intelligence, Surveillance, and Reconnaissance
- POC: Points of Contact
- PPOC: Principal Point of Contact
- S^3: Size, Speed, and Security
- S-KPP: Sustainment Key Performance Parameter
- SIDECAR: Semantically Informed, Dynamic Engineering of Capabilities and Requirements
- SoS: System of systems
- SWarF: Senior Warfighters' Forum
- TCAC: Technical Control and Analysis Center
- TEMP: Test and Evaluation Master Plan
- TEG: Tactical Exploitation Group
- TPC: Topographic Production Capability
- VAF: the Value-based Acquisition Framework for Evolutionary Information Systems
- UI: User Interface

Appendix B: Salient SIDECAR KB Statistics

While numbers (of terms, rules, etc.) are often misleading, it may be instructive to glance at the following statistics which characterize the "delta" – things we added to Cyc in the course of this project, the extensions to Cyc which, when loaded in on top of it, turn it into SIDECAR. Part of the final deliverable under our current contract will be a file of these new collections, individuals, predicates, rules, etc.

- New Cyc constants (terms) created: 453 (additions to the ontology)
- New Relations: 302
    - New Predicates: 266
    - Knowledge Acquisition Predicates: 138
    - New Functions: 36
- New Collections: 102
- New Individuals (Relations excluded): 96
    - New Microtheories (permanent): 11
    - KA Objects (fields, menus, dialog boxes, diagrams): 56
- Meaning Sentence Assertions: 37 (mappings to structured information sources)
- Rules for generating CDD Infrastructure: 91 (partial models of CDD sections)
- KB state-computing rules: 34
- KA Inference Rules (including KA object populating rules): 993
- KA assertion lifting rules: 483
- New assertions created in the JCIDS Document Theory: 2,974