Human/Autonomy Collaboration for the Automated Generation of Intelligence Products

Phil DiBona, Jason Schlachter
Lockheed Martin
Advanced Technology Laboratories
{phillip.j.dibona,jason.schlachter}@lmco.com

Ugur Kuter, Robert Goldman
Smart Information Flow Technology, Inc
{ukuter,rpgoldman}@sift.net

Abstract

Intelligence Analysis remains a manual process despite trends toward autonomy in information processing. Analysts need agile decision-support tools that can adapt to the evolving information needs of the mission, allowing the analyst to pose novel analytic questions. Our research enables the analysts to only provide a constrained English specification of what the intelligence product should be. Using HTN planning, the autonomy discovers, decides, and generates a workflow of algorithms to create the intelligence product. Therefore, the analyst can quickly and naturally communicate to the autonomy *what* information product is needed, rather than *how* to create it.

Operational Challenges for Intelligence Analysts

Intelligence analysts are responsible for the development of intelligence products in operational environments (OE) that continually evolve due to military, diplomatic, political, strategic, and tactical situations. The analysts in turn must maintain flexibility in their analytic tradecraft and adapt their analysis and work products to these dynamic situations. This analytic *agility*, one of the ten *Principles of Joint Intelligence*, is the ability to quickly shift focus and leverage skills and tools to address new problems, employing automated data handling and communications systems that are capable of responding to changing circumstances[1] . Analysts, therefore, need agile decision-support tools that can adapt to the evolving information needs of the mission, such as by allowing the analyst to pose novel analytic questions and to request automated monitoring for indicators & warnings on specific situations. Yet existing automation support tools fail to provide this level of support because they are often inflexible and cannot customize their execution to meet complex or novel situations. Therefore, the next generation of automation-support tools need to be driven by the analyst, providing the flexibility to select and tailor capabilities based upon the novel information needs of the current and anticipated mission state.

Composable Analytic Systems

A viable solution to address the agile information needs of analysts is the use of Composable Analytic Systems[2] (CAS). Composable software systems provide the capability to select and assemble components in various combinations to meaningfully satisfy specific user requirements[3]. The general goal of a composable design approach is to allow systems engineers to more rapidly investigate adapting existing technology to a

new mission problem[4]. It is this flexibility that makes composable designs attractive for intelligence analysis systems, allowing the extant enterprise services, algorithms, and data sources to be composed to meet novel and emerging mission information needs.

In many cases, composable systems are tools and environments that assist systems engineers in the design and deployment of larger, aggregate systems. However, the preferred creator of novel analytic queries and mission-information needs, the intelligence analyst is our target end-user for CAS, not systems engineers. Therefore, our investigation of CAS for mission analytics focuses on workflow systems, a particular class of composable systems. Workflow systems allow end-users (often without formal programming experience or training) to specify and execute an ordered set of computational tasks or processes usually in the form of an intuitive directed acyclic graph (DAG). A CAS capability allows intelligence analysts to allocate complex analysis, correlation, and fusion (ACF) tasks to autonomy, allowing the system to generate intelligence products that the human analyst may incorporate into their assessment of hypotheses.

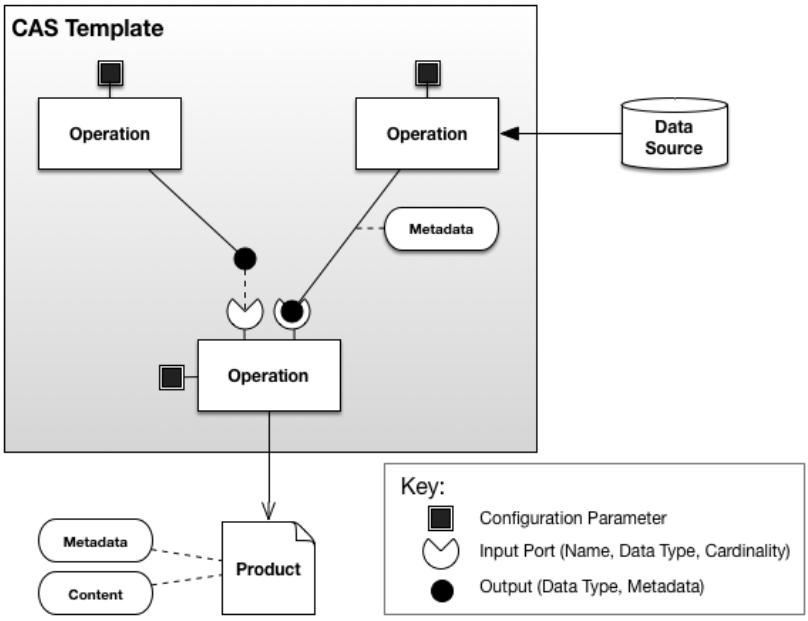

Figure 1: CAS Templates provide configurable processes to generate products

CAS Workflows, which we refer to as Templates, specify a configurable process in the form of a DAG (as shown in Figure 1), that is used to ingest one or more inputs (e.g., data sources, previously-generated intelligence products), execute a series of computational operations over that data, and generate a new product. The operations in the templates are cohesive, single-scope algorithms that perform a function across a wide breadth of capabilities, including: data I/O, data transformations, geospatial operations, data analysis, correlation, and fusion. Operations are comprised of exactly one output, zero or more inputs (from other operations), and zero or more configuration parameters that control its underlying algorithm's behavior. Operations must also expose specific metadata to the CAS infrastructure so that their name, description, and input/output specifications can be

understood by the end-users. The algorithm implementations managed within each operation may be deployed locally within the CAS execution engine, or it may be a remote enterprise service invoked by the CAS infrastructure. Having a large collection of enterprise-enabled operations provides the end-user with a virtual "toolbox" of operations that can be composed into a mission-focused template to execute complex behaviors that rapidly produce intelligence products that answer the end-user's analytic questions.

Example CAS Use Case

As an example, we shall discuss a fictitious scenario. Intelligence analysts have a time-critical hypothesis that the forces of an adversary nation, Pacifica, will want to defend a set of bridges in their country to ensure shipments of missiles. The intelligence cell, unfortunately, does not have an up-to-date database of Pacifica's infrastructure, including bridges. The first task for the CAS automation support tool is to identify a set of likely bridge locations. Then a second CAS task will examine recent Intel Reports about suspected adversary movements along major highway routes as well as Intel Reports of activity in the vicinity of those bridges, generating a list of likely bridges with adversary activity. Figure 2 shows a single CAS Template that performs these two analyses and helps the analyst answer the mission-focused question of "Where could the adversary be protecting missile routes?". In the template's first task, the analyst chose to find the geospatial intersection of bodies of water with roadways. In the second task, the analyst correlated these likely bridge locations with suspected routes and a set of Signals Intelligence (SIGINT) Reports, then create a heat map overlay that visualizes the likelihood of which bridges may be protected for missile movements.

In this simple example, the template can quickly generate an intelligence product in the form of a KML Map Overlay that identifies and filters hundreds of likely bridge locations that are correlated with potential adversary activity. The rapid generation of an intelligence product frees analysts from manual geospatial analysis and can answer mission-specific analytic questions.

LM ATL has conducted several small-scale evaluations of using CAS for providing automation support for Intelligence Preparation of the Operational Environment (IPOE) tasks. Qualitative feedback from analysts has indicated that CAS automation capabilities can reduce the amount of data gathering, filtering, and correlation work that is often manually done by analysts. By freeing up analysts from these complex and data-oriented tasks, they have more time for higher-level analysis of the generated products and for analytic collaboration with peers. Additionally, CAS provides analysts with an easier way to conduct "what-if" analysis by repeatedly executing CAS templates with varying configurations (e.g., resolution, areas of interest, changing features under analysis), providing a wider breadth of analyses against alternate and competing hypotheses.

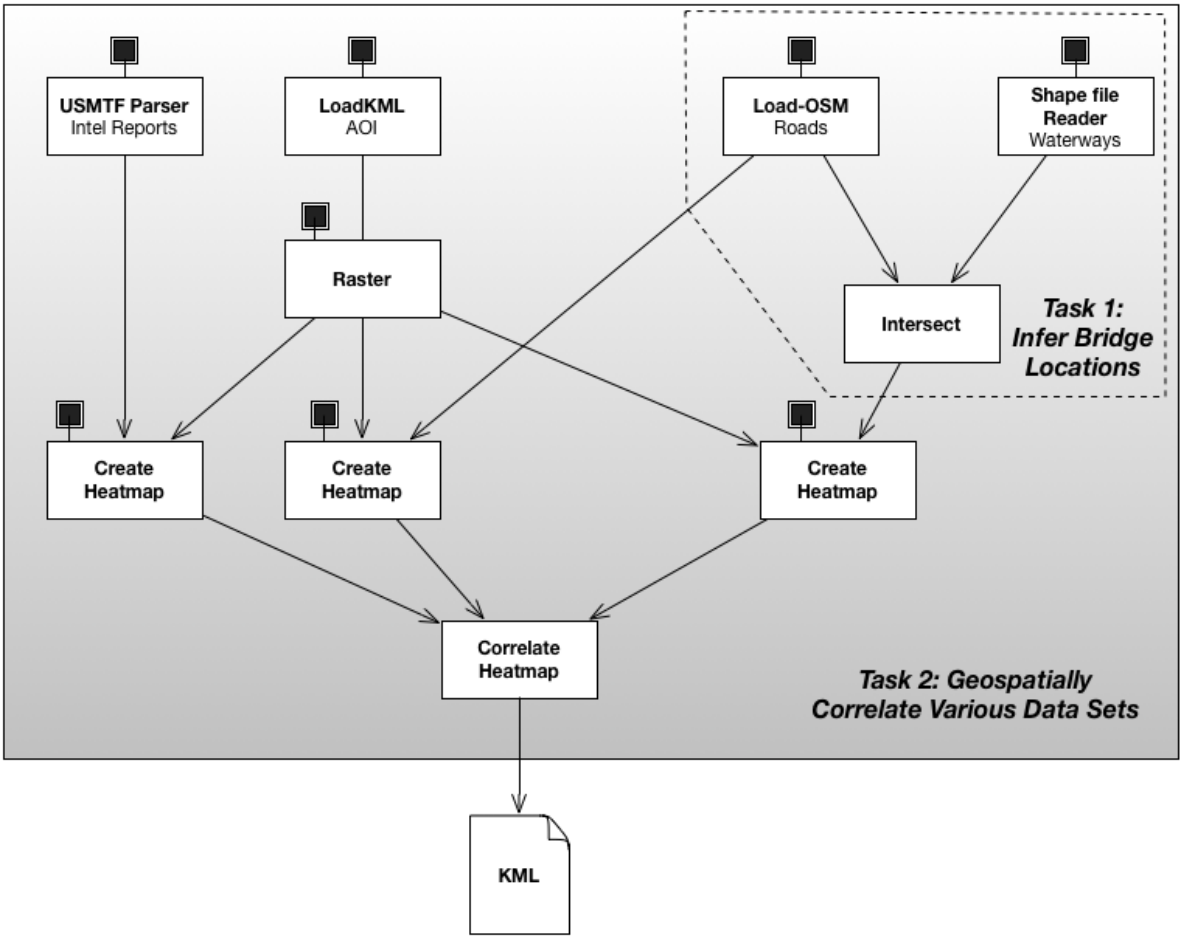

Figure 2: The example CAS template generates a product that may answer the analyst's mission-focused question

Operational Challenges for CAS

Despite these benefits, analysts cite specific challenges in adopting CAS into their operational workflow. Creating CAS templates require that the end users become experts on instructing the autonomy how to generate the desired work products. CAS templates can sometimes be very large and complex, as shown in Figure 2, and their success relies on ensuring the correctness of the specific details in the templates. These details may include (but are not limited to) ensuring transforms for incompatible data types and matching the correct data reader for specific data repositories. Therefore, template creators must specify how the template must generate the final products. *Power users* within the analysis cell can serve this function, but to realize the CAS vision of analyst-driven, mission-focused automation support, the individual analysts must be able to quickly and accurately compose their own templates.

In our Pacifica scenario, the analyst had to construct a template that set search limits onto the AOI of Pacifica, ingested specific geospatial data files (e.g., bodies of water, road networks), filtered out non-major highways, ingested Intel Reports, filtered those reports for recent SIGINT reports and adversary movements. Then the analyst instructed the template to build a heat map-style geospatial overlay that shows the correlation of information. While all of these tasks are composable and understandable by the analyst,

the process of template construction is very detail-oriented and requires knowledge of specific data sources, mechanisms to filter data with respect to the data schemas, and details regarding overlay construction. While the composition of such templates are feasible when created a priori, they are not readily composable during in-mission situations when the novel analytic questions are hypothesized.

This challenge motivates our CAS research toward a new goal, namely to provide analysts with the ability to rapidly specify to the computer system *what* intelligence product they need during missions, rather than constructing templates that specify *how* to generate the product a priori. To make this specification process more natural, quicker, and simpler for the analyst, we propose that the analyst specifies their product needs via a constrained English interface. With this user interface model, both the analyst and the autonomy are focused on the intelligence product, with the analyst specifying the product needed, and the autonomy satisfying that specification with the tools and data available.

The SWOOP System

Composable analytics can provide several benefits to information and intelligence analysts, but the challenges in creating such orchestrations inhibit operational adoption. We propose a capability that seeks to provide a semi-automated method of CAS Template creation that unburdens the end-user from designing the orchestration of CAS operations to generate intelligence products, while maintaining the CAS agility to answer novel analytic questions from the user. The SHOP2 Workflow Orchestration Planner (SWOOP) allows analysts to specify *what* the desired intelligence product should contain using a natural constrained-English interface, rather than creating a DAG that instructs the computer on *how* to create the same product. Once the product information needs are specified by the end-user using domain-specific terminology, SWOOP autonomously generates and executes a CAS Template that realizes that product specification.

SWOOP-enabled Example

In our CAS example scenario, the analyst constructed two templates that (1) identified likely bridge locations and (2) created a correlation of those bridge locations with likely mobile missile routes and SIGINT reports in the form of a heat map overlay. Using this same scenario, SWOOP will allow the end-user to generate the same products, but in a more natural constrained-English format that can be communicated quickly, as shown in Figure 3.

define new object DerivedBridges
  from intersection of roads, waterways
  in AOI Pacifica.

create heatmap
  from correlation of missile-routes, sigint, DerivedBridges
  in AOI Pacifica.

Figure 3: The analyst naturally specifies *what* product they want via a Constrained English interface

Internally, SWOOP parses this text, extracting the high-level tasks and goals, and automatically identifying the necessary algorithms/operations and generating a DAG that is topologically similar and functionally equivalent to the CAS template created by the user in the previous example. After post-processing, the resulting DAG is provided to the CAS engine, generating the product specified by the user. The DAG shown in Figure 4 illustrates a SWOOP-enabled CAS Template that is similar to the template in Figure 2.

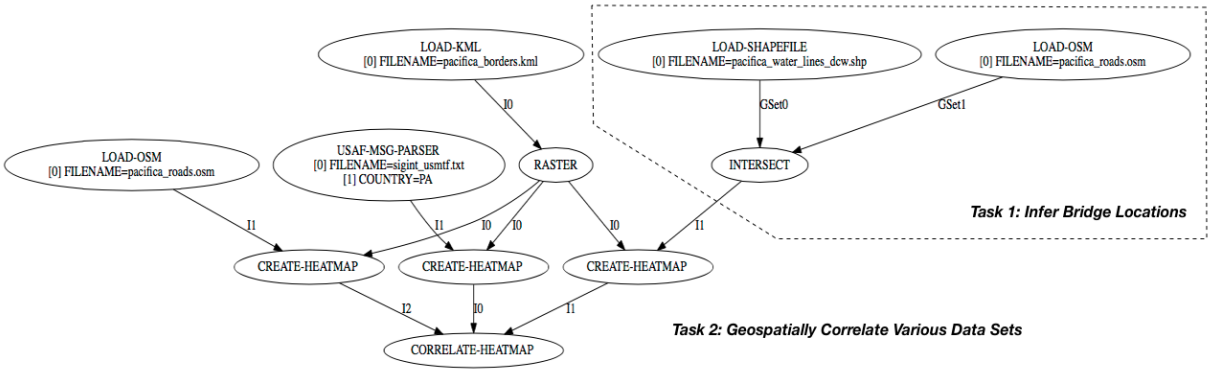

Figure 4: SWOOP generates an operation orchestration plan that is converted to a CAS template

SWOOP Architecture

SWOOP is comprised of several components and intermediate data items that enable its ability to auto-orchestrate algorithms and network services to generate intelligence products for its users, as shown in Figure 5.

The SWOOP process begins with the *Mission Context* data, an ontology that represents the information needed to bootstrap both the conversation with the user (e.g., Intelligence Analyst) as well as the planning subsystem. For the user's conversation, Mission Context provides the set of nouns and verbs available to the Constrained English for this particular installation of SWOOP. The nouns are the Areas of Interest, units, and data sources used to support the missions, while the verbs are mapped from the algorithms and enterprise services accessible by the CAS Engine. Additionally, the Mission Context provides the planning subsystem with a concept taxonomy specifying the breakdown of terms provided by the user and their mappings to core data types. The SWOOP Mission Context data is encoded in a multi-worksheet Excel spreadsheet for easy maintenance.

The *Constrained English Processor (CEP)* is the component that interacts with the user via a constrained natural language interface. CEP accepts the user's constrained natural language description of what the final Intelligence Product should be, then provides a validated set of product specifications to the *Mission Context Preprocessor (MCP)*. The MCP accepts the specification as well as the mission context then produces the lisp-style predicates required by the planner to generate a set of workflow plans.

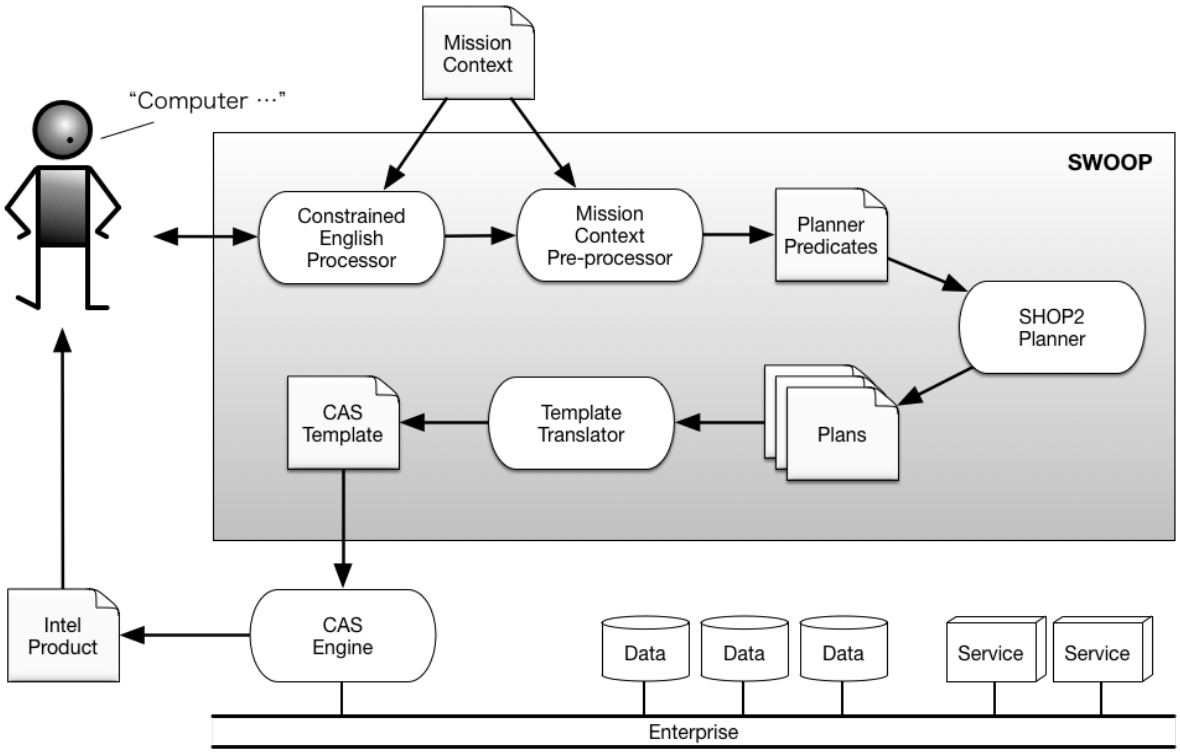

Figure 5: The SWOOP Architecture uses analyst and mission context to generate intelligence products

SWOOP uses the *Simple Hierarchical Ordered Planner (SHOP2)* domain-configurable automated planning system to generate a set of draft plans for the product specification provided by the user through the CEP and MCP. These plans are already in the form of directed acyclic graphs, but SHOP2 may produce multiple plans with small variations or alternate plans based on ambiguities of the mission context or product specification. The *Template Translator's (TT)* role is to accept these draft plans from SHOP2, validate their correctness, choose one, and transform it (syntactically) into a CAS Template. Once in the form of a CAS Template, the CAS Engine will instantiate and execute the orchestration of operations laid out in the SHOP2 plan, resulting in an intelligence product for the user.

The SWOOP Ontology

The SWOOP ontology (illustrated in Figure 6) contains all of the goals that represent the specifications of the user's desired intelligence product, as well as the contextual information that CEP needs to understand the verbs and nouns of the domain as well as information that SHOP2 needs to generate workflows for the user.

The *Concepts* are the hierarchical domain and mission-specific terms that the user will use. These may include: "transportation network" (and its sub-concepts "roads," "rails," and "airways"), "area of interest," "report" (and its sub-concepts "SIGINT", "HUMINT", etc). Concepts are abstract terms that represent information content, but are not directly indicative of any kind of data type or schema. The ontology allows for concepts to be mapped to *Types*. For example, the concept "road" would be mapped to a type of "line" indicating that roads are composed of a set of line strings. Just as concepts are hierarchical,

so are types. For instance, specific types of "line," "point," and "polygon" are all descendants of a generic type called "geospatial." A third class, *Data Format*, identifies the various schemas of information that the system can access/parse. This can include specific well-defined schemas such as Open Street Map (OSM) Textual Format[5], Geonames[6], or Keyhole Markup Language (KML)[7]. Data Formats should be as specific as possible so that SWOOP can match a "reader" operation to a schema. Having a Data Format of "XML" would be meaningless to the system without knowing the XSD or DTD that specifies the data being modeled. Therefore, Data Formats should identify a specific data schema when possible. *Data Sources* are the ontological instances that tell SWOOP exactly what data is available for use in the workflow, by specifying a location (e.g., URI, file path, database locator) of data that is of type *Data Type* and models a specific *Concept,* and constrained in time and/or space by zero or more *Constraints.* This is used to identify specific data sources available to the system such as: a file path for road networks in OSM format for the country of Pacifica and an HTTP URI for road networks in KML for only the southern portion of Pacifica. Between data sources and concepts, this ontology can provide a set of nouns for use by the CEP when interacting with the user.

The ontology also identifies the available *Operations* (i.e., software algorithms and services) that comprise the tasks in CAS Templates and can be executed by the CAS Engine. Operation instances can be one of two flavors: Data Source Operations or Data Transform Operations. Data Source operations are the software algorithms that understand how to read/parse/query the aforementioned Data Formats and output data into a CAS Type. Data Transform Operations are those software algorithms that ingest one more *Inputs* (*Outputs* of other operations) of specific CAS Types, and generate a new or modified data set of a specific CAS Type using the algorithm it wraps. The operations provide a set of verbs that are available for use in the CEP for interactions with the user.

For SHOP2 to plan an orchestration of operations that satisfy the user's specification of an intelligence product, the ontology must also be able to model that specification in the form of planning goals. The *Goals* of a planning scenario describe the desired concept(s), any spatio-temporal constraints that scope the goals, as well as any explicit methods (in the form of tailored operations) that must be used in the generation of the product. With the information modeled in this ontology, SHOP2 can generate a workflow of configured operations to: fetch raw data, transform data via select algorithms, and generate a product that satisfy the user's information needs.

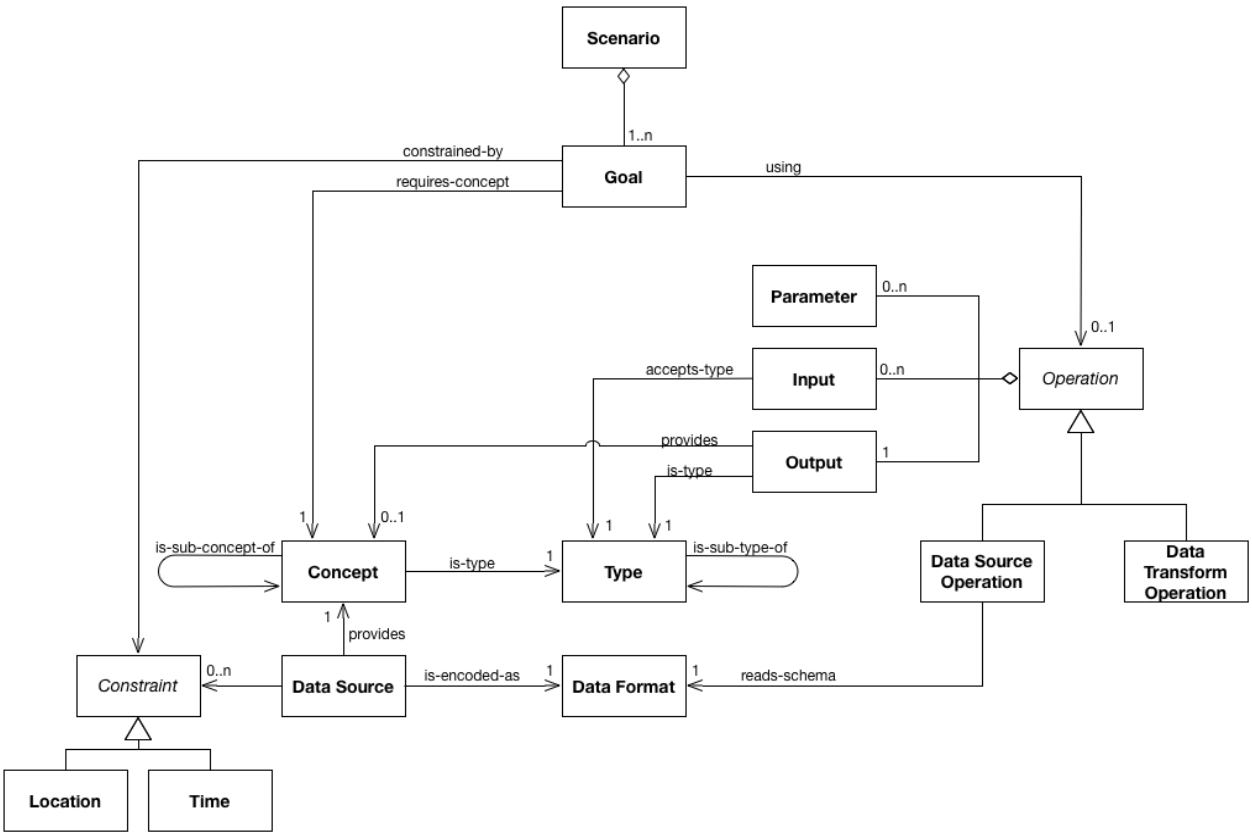

Figure 6: The SWOOP ontology facilitates human/computer collaboration by modeling both the mission and computing domains

Constrained English Processor

A Domain Specific language (DSL) is a computer language designed for use in a specific domain so that an expert may specify data models and processing instructions using familiar domain specific terminology and a simplified syntax. A major reason why DSLs may be preferred over a general-purpose programming language is they allow domain experts to develop, understand, and manage data models or processing steps directly. A DSL is defined by a grammar specification language, may have a custom syntax /context aware text editor, as well as several other components, such as a parser, linker, typechecker, and compiler used to automatically translate the domain experts' program into general purpose executable code such as Java.

We designed a prototype DSL that allows an analyst to *declare* that template outputs can be identified as new data types and features using structured natural language without having to specify details on *how* they will be created and from what data. The analyst's declarative specification is then parsed into a data structure representing a goal state for our planner which figures out the *how*.

See Figure 3 for an example specification of a new data type, *DerivedBridges*. Here we define a bridge object anytime we find in our dataset an intersection of a ground transportation route (road, rail, etc.) and a waterway (river, lake, etc.) where there is not already a structure labeled *Bridges*. We also impose additional constraints, such limiting

our search to a particular Area of Interest (AOI), as done in Figure 3, where we limit the definition to Pacifica. This specification will be converted into executable code by our custom-built parser, linker, typechecker, and compiler, and passed off to our planner so that it may find the right sequence of actions to take to create these types of objects.

We leverage Xtext[8], an open source language specification framework that provides a DSL grammar and allows for easy creation of a context aware IDE as well as the parser, linker, typechecker, and compiler. Ultimately, Xtext builds an internal data structure by parsing the DSL specification and executes code we have created to query that data structure and to write new code based on that structure. In our case, we create a goal state specification to pass into our planning algorithms.

Mission Context Pre-processor

The purpose of the Mission Context Pre-processor is to provide a *Scenario* (as specified in the SWOOP ontology) in the form of planning predicates, to the SHOP2 planner. This scenario is a self-contained set of planning goals from the user and the context that includes all known data sources, concepts, and CAS operations. MCP receives the scenario goals from CEP, aggregates this Scenario instance from various static and dynamic sources, validates the scenario, and then generates the lisp-style input to SHOP2. A subset of an example scenario is shown in Figure 7.

The Simple Hierarchical Ordered Planner (SHOP2)

SHOP2, a general-purpose hierarchical task network (HTN) planner[9][10] that takes mission specifications in terms of goals, mission context, and priorities, and refines them into detailed plans. Unlike still actively-used HTN planners[11][12][13][14][15][16], SHOP2 can operate either wholly autonomously, or as part of a mixed-initiative human/automation team. When running fully automated, SHOP2 can autonomously choose methods for achieving tasks and subtasks, and assign resources to those tasks and subtasks. Alternatively, the human operator can "dive down" into the hierarchical structure of the play to offer increasingly specific instructions about exactly how a given task must be performed, or which resources must be assigned to it.

Like other HTN planners, and unlike first-principles planners, SHOP2 searches top-down from a task specification or a network of tasks, rather than chaining together primitive actions. SHOP2 reasons about priorities to generate a plan, re-planning whenever the goals and priorities evolve or the situation changes. SHOP2 decomposes complex mission tasks into more primitive tasks and sub-tasks (methods), thus building a plan tree that terminates at leaves corresponding to primitive "actions" (operators). Methods describe how to perform complex tasks. A method definition associates a task with a set of preconditions and a task network. When the preconditions are satisfied, a task that matches the method definition can be decomposed to the given task network. Task networks are workflows of tasks that may be constrained to be ordered, or that can be executed in any order (unordered). For example, the HTN method shown in Figure 8 is written for CAS templates in SWOOP.

```
;; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Type Hierarchy
(subtype-of geospatial lines)
(subtype-of geospatial polygons)
(subtype-of geospatial points)
(subtype-of geospatial cells
;; ...

;; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Concept Hierarchy
(subconcept-of transportation roads)
(subconcept-of transportation airways)
(subconcept-of usmtf-report sigint-report)
;; ...

;; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Concept-Type Maps
(concept-type-map roads lines)
(concept-type-map nais polygons)
(concept-type-map heatmap cells)
;; ...

;; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Goal(s)
(
 (find
  (
   ;; Task #1:
   (intersect
    (:aoi pacifica)
    (:concept (roads waterways))
    (:as DerivedBridges)
    )

   ;; Task #2:
   (fuzzy-and
    (:aoi pacifica)
    (:concept (missile-routes DerivedBridges sigint))
    (:as heatmap)
    )
   )
  )
 )
```

Figure 7: The SWOOP scenario provides mission context to the automated planner

SHOP2 is a planning system that "fills in the details" of the given mission specification and intent, computing plans to accomplish tasks within specified constraints. SHOP2 takes procedural and operational descriptions of missions, analyst heuristics, commander's intent, and planning knowledge, and refines them into detailed plans. Plans can contain hierarchically defined sub-goals that are assigned to lower-level units and trigger local, context-sensitive planning and execution behaviors based on local, possibly incomplete or uncertain information. This is possible because SHOP2 performs task decompositions in

the order those tasks will be executed in the CAS engine. As a result, SHOP2 knows the state of the world at all times using this forward-chaining search paradigm; it enables SHOP2 to perform axiomatic inference, arbitrary numeric computations, incorporate optimization during planning, call arbitrary external functions and programs. For instance, in the above example, it is possible to configure INTERSECT because SHOP2 already know AOI information and target concepts from the evolution of the planning state through other computations and planning actions until when this method is used.

```
(:method (intersect ?template-id ?inputs ?output
                    ?output-concept)
  ;; Preconditions
   ((task-constraint ?template-id :aoi ?user-aoi)
  (task-constraint ?template-id :concept ?user-concept-list)
  (task-constraint ?template-id :as ?user-target-concept)
  (map-to ?user-concept-list nil ?user-types)
  )
    ;; Subtasks:
   (
  (intersect-inputs ?user-concept-list ?user-aoi ?user-types
                    0 nil ?input-triples)

  (intersect-output ?intersect-id ?input-triples
                    ?output ?output-type ?user-target-concept)
  ))
```

Figure 8: HTN methods associate a task with preconditions and a task network

Configurability of the HTN methods in SHOP2 is not only over the inputs and outputs of the subtasks. SHOP2 also allows HTN methods to be written abstractly and configurable for the subtasks that will accomplish. For example, in SWOOP, the input DSL could specify different workflow tasks with different configurations. From knowledge-engineering and usability perspectives, it is not reasonable to encode every possible workflow task and its configuration that can be specified by the user as an HTN method. Instead, the HTN method shown in Figure 9 controls this abstract input and its potential configurations. This method takes the tasks that the FIND directive from the user as an input, as opposed to hard-coding every combination them in the subtasks. The first and third subtasks of the method creates and closes the XML document whose content will encode the workflow generated by the planning process. The subtask ACHIEVE-TASK iteratively walks over the configuration preferences in the parameter ?TASKS and accomplish each of them as it stiches the solution workflow.

```
(:method (find ?tasks)
   ((template-header ?xmlns ?xmlns2 ?xmlns3 ?exec-interval
                     ?exec-iter ?user-template ?user-id ?user-name
                     ?template-id ?template-name ?template-descr
                     ?template-catID))

   (
    (!!start-xml-template ?xmlns ?xmlns2 ?xmlns3 ?exec-interval
                          ?exec-iter ?user-template ?user-id ?user-name
                          ?template-id ?template-name ?template-descr
                          ?template-catID)
    (achieve-task ?tasks ?template-id)
    (!!close-xml-template ?template-id ?template-name)

    ))
```

Figure 9: Configurability of HTN methods allow for generalization across a wide range of tasks

Template Translator (TT)

SHOP2 produces a set of predicates that represent a directed acyclic graph (DAG) of CAS operations. This DAG provides enough details to ensure that the outputs of operations are routed to the correct input ports of downstream operations (as shown in the edge labels in Figure 4) as well as the configuration parameters for the operations (numbered text inside nodes) based on the scenario predicates provided by the MCP. Because the DAG plan produced by SHOP2 is functionally equivalent to a CAS Template, translating the output is a trivial step. However, before the DAG can be translated, the Template Translator (TT) has the important task of choosing the output plan to translate. SHOP2 can provide multiple plans based on the specificity of the planning goals and also the HTN methods that were used in the creation of the plans. TT first confirms that the DAGs are indeed valid, then removes from consideration any that may have erroneous or missing components. SWOOP will employ one of two strategies to select the DAG for conversion to a CAS Template: *Most Confident* and *Smallest Depth*. SHOP2 provides its results in descending order of confidence, and TT can select the first valid DAG. Typically, however, we have found through testing various scenarios that choosing the DAG with the smallest tree depth gives the DAG that closest to optimal. For future work, we plan to research other methods for DAG selection, including plan critique methods and machine learning approaches.

Conclusions

We have created a prototype of the SWOOP capability, building upon our experiences in CAS and automated HTN planning. We envision that SWOOP can be used effectively in environments where information analysts acquire, filter, transform, analyze, correlate, and fuse large quantities of Multi-INT or heterogeneous data, such as intelligence analysis and Processing, Exploitation, and Dissemination (PED) cells within the Intelligence Community.

The SWOOP capability focuses on our proactive human/autonomy collaboration approach[17] where both the human and the autonomy contribute to the same work product. In this case, the work product is an intelligence product specified by the analyst, where the autonomy acquires, filters, correlates, and fuses information for that product, facilitating the human user in maximizing her time analyzing and drawing conclusions from the autonomy-provided information. By allowing the analyst to specify the information needs via a natural language interface using domain-specific terminology, SWOOP minimizes adverse effects on the analyst's workflow.

We have identified a set of areas in which SWOOP can be improved where further research can improve its effectiveness in operational environments. Firstly, the SHOP2 planner can provide multiple plans from a single user product specification. The accuracy of the SWOOP-generated CAS templates will benefit from a more robust method of validating, critiquing, and selecting the best plan that meets the user's information needs. Another area for continued research is automated learning and/or generation of the HTN methods used by SHOP2. Currently, prior to use, some HTN methods require that a human user identifies and encodes common CAS template design patterns before they can be used by analysts in their natural language product specification. By automating this process via machine learning would provide greater flexibility of the SWOOP capability.

References

[1] "Joint Publication 2-0, Joint Intelligence". *Defense Technical Information Center (DTIC)*. Department of Defense. 22 October 2013. pp. II:10-11. Retrieved January 16, 2017.

[2] P. DiBona, , J. Llinas,, and K. Barry. "Composable Analytic Systems for next-generation intelligence analysis", Proc. SPIE 9499, Next-Generation Analyst III, 94990N (May 15, 2015); doi:10.1117/12.2184177; http://dx.doi.org/10.1117/12.2184177

[3] Davis, Paul and Committee on Modeling and Simulation for Defense Transformation, National Research Council. Defense Modeling, Simulation, and Analysis: Meeting the Challenge. Washington, D.C.: National Academies Press, 2006.

[4] Oster, Christopher, Wade, Jon, Ecosystem Requirements for Composabilty and Reuse.

[5] OpenStreetMap. OSM File Formats. Retrieved on February 22, 2017 from http://wiki.openstreetmap.org/wiki/OSM_file_formats

[6] Geonames worldwide data set. Retrieved on February 22, 2017 from http://geonames.org, This work is licensed under a Creative Commons Attribution 3.0 License.

[7] Open Geospatial Consortium. Keyhole Markup Langauge specification. Retrieved on February 22, 2017 from http://opengeospatial.org/standards/kml

[8] The Eclipse Foundation. Xtext. Retrieved on February 22, 2017 from http://www.eclipse.org/Xtext/documentation/index.html

[9] T.-C. Au, U. Kuter and D. Nau. 2005. Web Service Composition with Volatile Information. Proceedings of the ISWC-05.

[10] D. Nau, T.-C. Au, O. Ilghami, U. Kuter, H. Munoz, W. Murdock, D. Wu, and F. Yaman, 2005. Applications of SHOP and SHOP2. *IEEE Intelligent Systems*. 20(2):34-41.

[11] Tate, A. (1977). Generating project networks. In IJCAI-77, pp. 888–893.

[12] Tate, A., Drabble, B., & Kirby, R. (1994). O-Plan2: An Architecture for Command, Planning and Control. Morgan Kaufmann.

[13] D. E. Wilkins, Practical Planning: Extending the Classical AI Planning Paradigm, Morgan Kaufmann, San Mateo, CA, 1988.

[14] Erol, K., Nau, D., & Hendler, J. (1994). HTN planning: Complexity and expressivity. In AAAI-94.

[15] Erol, K., Hendler, J., & Nau, D. (1996). Complexity results for Hierarchical Task-Network planning. Annals of Mathematics and Artificial Intelligence, 18, 69–93.

[16] Sohrabi, Shirin, and Sheila A. McIlraith. "On planning with preferences in HTN." Proc. of the 12th Int'l Workshop on Non-Monotonic Reasoning (NMR). 2008.

[17] P. DiBona ; A. Shilliday and K. Barry "Proactive human-computer collaboration for information discovery ", Proc. SPIE 9851, Next-Generation Analyst IV, 985102 (May 12, 2016); doi:10.1117/12.2222805; http://dx.doi.org/10.1117/12.2222805