

# Coordinated Data Acquisition on Sensor Webs

Robert Morris<sup>1</sup>, Jennifer Dungan<sup>1</sup>, Petr Votava<sup>1,2</sup>, and Lina Khatib<sup>1,3</sup>

(1) NASA Ames Research Center

(2) California State University, Monterey Bay

(3) Perot Systems Government Services

## Abstract

An Earth-observing sensor web is an organization of space, airborne, or in situ sensing devices for collecting measurements of the Earth's processes, thus promoting the interdisciplinary study of the Earth system. Potential users of a sensor web, including Earth scientists and authorities responsible for environmental monitoring, disaster mitigation and management, require a centralized means to effectively access sensing resources. A number of approaches to effectively manage a sensor web to ensure useful coverage and coordination have been proposed. This paper focuses on one aspect of sensor web coordination related to the formulation of Earth science goals and their transformation into requests for sensor web services. Automating parts of this process using recent advances in intelligent control software technology will offer improved sensor web effectiveness. In this paper we describe the needed capabilities and show how these capabilities can be integrated into a full sensor web management system.

## Introduction

An Earth-observing sensor web is an organization of space, airborne, or in situ sensing devices for collecting measurements of the Earth's processes, thus promoting the interdisciplinary study of the Earth system. Referring to this collection as a web suggests both coverage, i.e., the collection is distributed over a broad area for long periods of time; and coordination, i.e. sensing activity between distinct sensors in the collection can be linked together in location or time. A number of approaches to operating a sensor web to ensure useful coverage and coordination have been proposed.

Direct management and control of the resources on a sensor web are distributed. For example, daily scheduling of observations on the ETM+ sensor on Landsat 7 is managed by the Landsat science team and mission operations. Users of ETM+ data typically access the sensor by submitting requests for data stored in the Land Processes Distributed Active Archive Center (LP DAAC). More rarely, a user may submit request that the sensor be retargeted for acquiring future data. Either way, there are layers of control and management of sensing resources that are accessed through the submission of requests.

Potential users of a sensor web, including Earth scientists and authorities responsible for disaster monitoring, mitiga-

tion and management, require a centralized means to effectively access sensing resources. Efforts such as the OGC's Sensor Web Enablement (SWE) activity (?) seek to provide the interfaces and protocols for centralized access to a sensor web. To be effective, it should be possible for a user to be able to describe a desired data product without worrying about the details of where and how to retrieve the data, somewhat in the same way a user of a relational database uses SQL to describe a desired set of tuples without describing the physical location of the data.

The research in this paper describes information technology that will allow users to formulate goals in terms of a set of spatial, temporal, and resource constraints. These goal specifications will then be automatically transformed into sequences of requests for data that will satisfy the constraints. These sequences will be expressed in a language that can be executed autonomously. This technology will improve access to the sensor web by providing a layer of automation between the formulation of Earth science goals and the acquisition of the required data.

The key to the proposed technology approach is the idea that a sensor web can be viewed as a complex *controllable* physical system. From this viewpoint it is then possible to leverage recent advances in *autonomous control* technology to automate the process of accomplishing Earth science goals by *reconfiguring* the resources for acquiring, storing and analyzing data. The remainder of this paper describes this approach in detail. First, we define the key notion of goal-directed data acquisition. Then we describe the architectural components of an automated goal-directed data acquisition system based on the model of the sensor web as a complex control system. Finally, an implementation of the components of the architecture will be described.

## Example

Studies of Earth processes typically require data that are distributed in space and time, requiring the collection of multiple data sets from multiple sensors. For example, consider the following simple request for cloud-free GOES data:

*The goal is to acquire the minimum number of GOES data throughout a day for a period of a month while maximizing the cloud-free data. To ensure this, the National Digital Forecast Database (NDFD) forecast on cloud cover will be acquired. This forecast is*

*produced every 3 hours and forecasts the data out to 7 days; we will acquire the forecast once a day. If the sky is expected to be clear, obtain GOES data only per every 3 hours during the daylight hours. With higher probability of clouds in the forecast, the acquisition of GOES data will be every 15 minutes in order to maximize the probability of obtaining a clear view.*

Accomplishing the goal described in this scenario using current sensor web information technology requires that a user frequently formulate and submit requests for data. The user requires detailed knowledge of protocols and locations for submitting requests. Although the OGC-SWE effort will provide the infrastructure for centralized access to distributed data, the user, in general, will still be required to plan and execute the sequences of requests for accomplishing their goals.

### Goal-directed Data Acquisition

The overall objective of an Earth observation *campaign* is to increase understanding and reduce uncertainty in knowledge of the Earth's processes. The results of campaigns are used to advance science (initialize or validate Earth process models) or may be used directly in near real time for decision-making (e.g., hazard mitigation). The output of a campaign is one or more *data products*.

A campaign can be defined in terms of one or more goals. Types of goals include

- Characterize/classify: obtain or identify the values of a quantity;
- Monitor: watch for a significant change or for the threshold of some quantity to be reached;
- Compare: determine the similarities and/or differences between or among several instances of the same process or quantity;
- Validate: determine whether a prediction made by a model is correct or whether observations from a sensor are accurate;
- Predict: determine when a specific event will happen or how a quantity will evolve in the future;

Characterization is the most basic and common of goal types. Monitoring may be considered as a series of characterizations over time. A validate goal can be considered a special case of a compare goal, as the former typically involves comparison of models and measurements. Goals may involve direct observations and/or Earth system models or their components. Prediction especially always involves the use of a model.

An Earth observation goal is a specification of desired data products. A campaign consists of a set of goals that can be related in different ways. In addition to goals, a campaign may consist of other "supporting activities" required to accomplish goals. A supporting activity typically involves the transformation of data into a form that can be used as input to accomplish a goal. For example, in the GOES scenario a cloud-cover forecast is transformed into a quantity designating the frequency with which GOES data are obtained. The

transformation is an example of what we call a supporting activity.

A goal can be defined by specifying constraints on the following attributes:

- *what* is to be measured;
- *where* the measurement(s) are to be taken;
- *when* the measurement(s) are to be taken; and
- *how* the data are acquired, i.e., the resources to be employed.

Goals specify products that arise from three potential resources: directly from sensors, from data archives, or from models. For example, a goal might include a request that time be allocated on a specific sensor in the future. More commonly, goals request access to data already acquired and stored in an archive. Finally, the data product specified by a goal may require the execution of a model.

A data product may consist of a collection of similar products that are *distributed* in space or time. A *distribution* of a data product consists of the number of products collected and their "spacing" in space and/or time. For example, "once a week for 10 weeks" can describe a distribution of a product: it implies 10 instances of a product each separated by one week.

In general, the accomplishment of one goal might depend on the output of another goal. In the GOES scenario above, the temporal distribution of the GOES data acquisition is determined by a cloud cover forecast product, which forms a separate goal. This is an example of what we have called "model-based observing" (?). Similarly, the spatial distribution of a set of observations of a pollution plume may depend on the predictions by models of where the plume is expected to be over time. Here the model determines the spatial distribution of the observations.

In the example above, there are two goals: a characterize goal to *acquire GOES data* and a predict goal to *NDFD forecasts*. There is also the supporting activity of transforming the NDFD forecast into an updated temporal distribution for acquiring GOES data. The campaign is depicted visually in Figure ???. The figure shows the goals and supporting activities and also shows that the goals depend on product web resources. Again, this scenario is an example of model-based observing, insofar as part of GOES data goal (viz. the temporal distribution) depends on the output of a model.

### Campaign Specification, Generation and Execution

The previous section characterized goal-driven sensor web activity as a campaign, where a campaign is a sequence of goals of certain basic types, consisting of a set of constraints on where, when and how the products are to be produced. This section describes in more detail the process by which goals can be transformed automatically into sequences of sensor web requests for accomplishing them.

Data from Earth observing platforms are continually processed and stored. Campaign goals are accomplished by a sequence of sensing, storing and processing actions. Consequently, a more complete description of the resources used

to accomplish campaign goals will include, in addition to sensors, data archives and the models and other processing elements for producing processed data products.

Abstracting the goal-directedness of Earth observation campaigns allows a distinction to be drawn between three activities:

1. Campaign goal formulation;
2. Planning for accomplishing the campaign; and
3. Execution of the plan.

In our view, goal formulation is primarily a human activity, planning is an activity that involves a mix of human and automated capabilities, and plan execution is something that can be to a large extent automated.

### Goal formulation

Based on the discussion in the previous section, campaign formulation is a process of identifying the campaign goals, supporting activities, and constraints and inter-dependencies associated with them. Some language for defining goals and associated constraints and dependencies is therefore required. Figure ?? shows the GOES scenario arranged as a tree, with two data acquisition goals (get forecast and get GOES data) and one processing goal that takes the forecast data and transforms them into a temporal distribution for acquiring GOES data. Leaves of the tree are labeled with the spatial and temporal attributes of the goals. Horizontal arrows depict dependencies or other constraints between the goals. For example, there is a dependency represented by the arrow between the Cloud Forecast leaf node (depicting the output of that acquisition) and the left child of the Frequency process node (representing the input to that process). Similarly, there is a dependency between the output of the process that computes the frequency of getting the GOES data and the GOES acquisition goal itself. There are also links representing the constraint that the forecast data cover the same region and time as those of the GOES acquisitions.

### Planning

The planning process provides the means of transforming the goal description into a sequence of sensor web activities that satisfy the goal constraints. Furthermore, because we want the campaign to be executed autonomously, the plan should be stated in a language that can be interpreted and executed as a set of sensor web requests.

The transformation of goals into executable request sequence requires two models. First, a *campaign model* describes the decomposition of each campaign goal into a sequence of simple activities. We have identified two simple activities in a campaign: the acquisition of data and the processing of data. Each campaign can be depicted as a sequence of one or the other of these simple activities. For example, in Figure ?? the GOES campaign is depicted as a sequence of two data acquisitions and a single processing activity. Horizontal arrows represent dependencies among the activities, whereas vertical arrows depict acquisitions from the sensor web (through archived data).

Secondly, a *sensor web model* describes the capabilities and constraints associated with different resources on the

sensor web. This model is used by the planner to ensure that requests submitted to a resource are “appropriate”, i.e., can be feasibly serviced by that resource, have the proper format, and adhere to other constraints that the resource may impose. For example, a resource model will specify that accessing a particular archive will require a request be submitted with a certain format, followed by an acknowledgment of receipt of the request no later than a certain time after the request, followed eventually by a receipt of a (pointer to) the requested data.

## Implementation and Integration

As a testbed for developing the concepts outlined above, we are implementing an architecture for coordinating sensor web activities based on four layers:

- A *planning layer* for transforming goals into executable request sequences;
- An *execution layer* of “web managers” for executing the sequences;
- A *service layer* for providing the protocols and standards for sensor web access; and
- A *sensor web layer* providing a testbed for simulating coordinated web activity.

The layered architecture is illustrated in Figure ?. The top layer includes the software components for translating campaign goals into executable plans. The Web Manager Layer contains software systems for executing the plans. The OGC layer provides the interfaces to the sensor web. As a sensor web example, TOPS, described below, we include the data repositories and models, as well as the sensors themselves. In the discussion that follows, we describe key software components that will be applied on each layer.

### Planning

In the sensor web domain, planning consists of the process of transforming a specification of goals, constraints and dependencies into a sequence of sensor web requests or processing actions. These sequences can include conditional or iterative behaviors. A concise representation of sensor web actions and the operators for composing sequences that satisfy the campaign goals can be based on finite automata. For example, consider the simple action *Repeatedly get the latest NDFD forecast*. This can be viewed as waiting for a condition (call it *changed*) to become true, indicating that the latest forecast is available, followed by a command to fetch the data. In the format of the Labeled Transition State Analyzer (LTSA) (?), this sequence is depicted as follows, using a process algebra syntax called Finite State Processes:

```
Get_NDFD_forecast =  
    (read.forecast['changed'] ->  
      command_get_forecast -> END).
```

The labels indicate state, and the arrows transitions. This example can be viewed as an instantiation of a pattern that is repeated for any sensor web action that involves waiting for the latest data from some model to be available for retrieval. These patterns become the basis of the model for

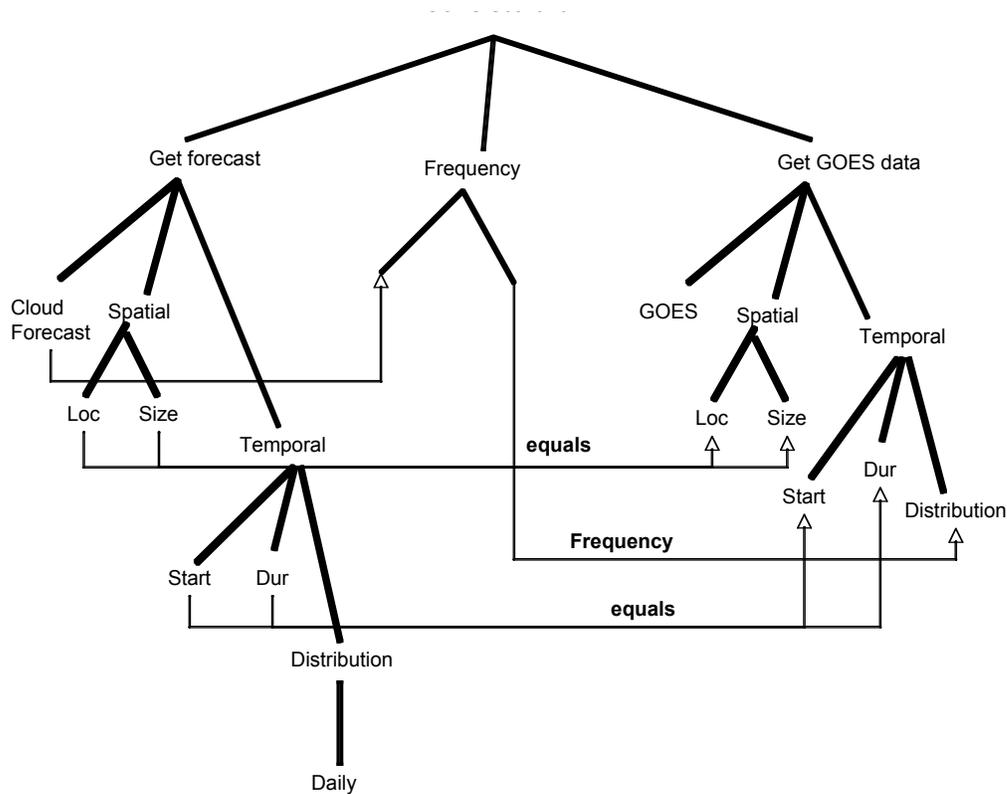


Figure 1: The GOES campaign goals arranged as a tree with dependencies

all goal-directed campaign plans. That is, all campaigns can be viewed as a set of ordered instantiations of such patterns, where the instantiations and orderings arise from the constraints and dependencies specified in the goals. LTSA allows for the rapid design and verification of such planning models, as well as a mechanism for translating the plan into an executable format in a language called PLEXIL (described below). For example, the FSP program

```
Seq = (read.sky[sv:SkyValues] ->
  if (sv == 'clear') then
    Get_GOES_data_clear[180]
  else Get_GOES_data_clear[15]),
Get_GOES_data_clear[freq:FreqVal] =
  (read.time[freq] ->
    command_acquire_goes_image ->
    Get_GOES_data_clear[freq]).
|| GOES_Plan = (Get_NDFD_forecast || Seq).
```

represents the GOES data acquisition plan as a sequence with two parts: getting the forecast, and getting the GOES data, with a frequency depending on the result of the model prediction. In addition to providing a concise representation of the campaign model used in planning, the LTSA tool allows for automated verification of the plans generated from

the model.

### PLEXIL

Plan Execution Interchange Language (PLEXIL) (?) is a language for designing autonomous execution systems. Control is specified as a set of execution nodes, arranged in a hierarchy, where leaf nodes are command invocations. Attached to each node are conditions that drive node execution. The Plexil Universal Executive interprets a Plexil representation of an execution control instance. Plexil also allows for monitoring resources and the status of executing commands.

The fundamental building block of a PLEXIL plan is a node, associated with a set of conditions that must be true for the node to execute and content that describes what gets executed. Nodes are arranged in a tree structure, representing different levels of abstraction in the plan. At the leaves of the tree are nodes that perform primitive actions, including commands to the system being controlled. In our domain, leaf nodes correspond to processing nodes or nodes for acquiring data from the sensor web.

PLEXIL plans can be written in a number of ways including directly in XML and in more concise languages, using syntactic enhancements. Figure ?? shows a PLEXIL plan

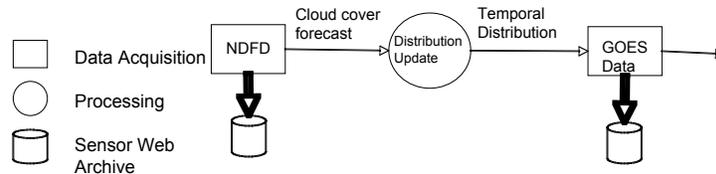


Figure 2: GOES scenario structure. Arrows indicate dependencies among activities.

for the example GOES scenario written in the functional language LISP. The plan has a set of nodes which include two command nodes for requesting forecast data and GOES data, as well as nodes for retrieving the data and corresponding to the node for processing forecasts into a frequency for acquiring GOES data.

### TOPS

The Terrestrial Observation and Prediction System, TOPS (?), is a data and modeling software system designed to seamlessly integrate data from satellite, aircraft and ground sensors, and weather/climate models, with application models to quickly and reliably produce operational nowcasts and forecasts of ecological conditions. Through automation of the data retrieval, pre-processing, integration, and modeling steps, TOPS is able to reliably provide data on current and predicted ecosystem conditions, allowing TOPS data products to be used in an operational setting for a range of applications. The TOPS system currently holds about 8TB of data on-line. The acquisition of satellite, ground station and model data is automated and the data are obtained in periodic intervals ranging from 15 minutes to several months.

TOPS has been engineered to automatically ingest various data fields required for model simulations. Ingested data go through a number of preprocessing filters, streamlining the input data to facilitate the simulations. After passing through a specification interface in which each parameter is mapped to a list of attributes (e.g., source, resolution, quality), each data field is self-describing to TOPS component models such that any number of land surface models can be run without extensive manual interfacing. Similarly, the model outputs also pass through a specification interface facilitating post-processing such that model outputs can be presented as actionable information, as opposed to just another stream of data.

### OGC

The Open Geospatial Consortium's (OGC) Sensor Web Enablement (SWE) activity is establishing the interfaces, standards and protocols for a centralized access to the sensor web (?). SWE is divided into a number of components, each of which contains models, services or XML encodings of various aspects of the sensor web. For example, the Sensor Model Language (SensorML) contains models and encodings for sensors, and the Observations and Measure-

ments component contains the same for sensor observations and measurements. These models and encodings provide an implementation of the sensor web model described above. Among the relevant services, the Sensor Observation Service (SOS) allows for requests for observations to be submitted, the Sensor Alert Service (SAS) provides an alert and notification mechanism to specify how alert or "alarm" conditions are defined, detected, and made available, and the Sensor Planning Service (SPS) determines the feasibility of a desired set of requests.

### Future Work and Conclusion

The focus of this paper has been on the representation of Earth science campaigns for sensor web coordination. Two important capabilities not discussed here are the tools for goal formulation and plan execution.

A measurement or processing goal requires specifying a set of values for features related to the location, time, and measurement type. As noted in the simple scenario we've used throughout the paper, some of these features might not be known at goal formulation time and may be dynamically generated during plan execution through model-based observing. In other cases, however, the features need to be discovered before an initial plan can be generated. For example, the user might require additional information about sensor web resources, e.g., about whether the needed data exist or where they are located. This step of *pre-planning for goal formulation* can be partially automated through the use of so-called sensor web discovery services. Again, the OGC effort offers services to support this activity. Future work will develop an extension to the planning language and interface capabilities to enable the use of discovery services for pre-planning during goal formulation.

Executing a plan in a changing environment requires *robustness*, an ability to respond to threats to the accomplishment of the plan's goals that emerge from these changes. This capability is important in the sensor web domain for a number of reasons. First, there is uncertainty in the sensor web domain due to the distributed nature of sensor web control. As noted above, control of resources in a sensor web is distributed, sensors may be oversubscribed (i.e., there may be more requests for a sensor than can be serviced), and requests for access to a resource may be rejected. Second, the Earth process under investigation is not controllable and may behave in ways not predicted by any model. Plan exe-

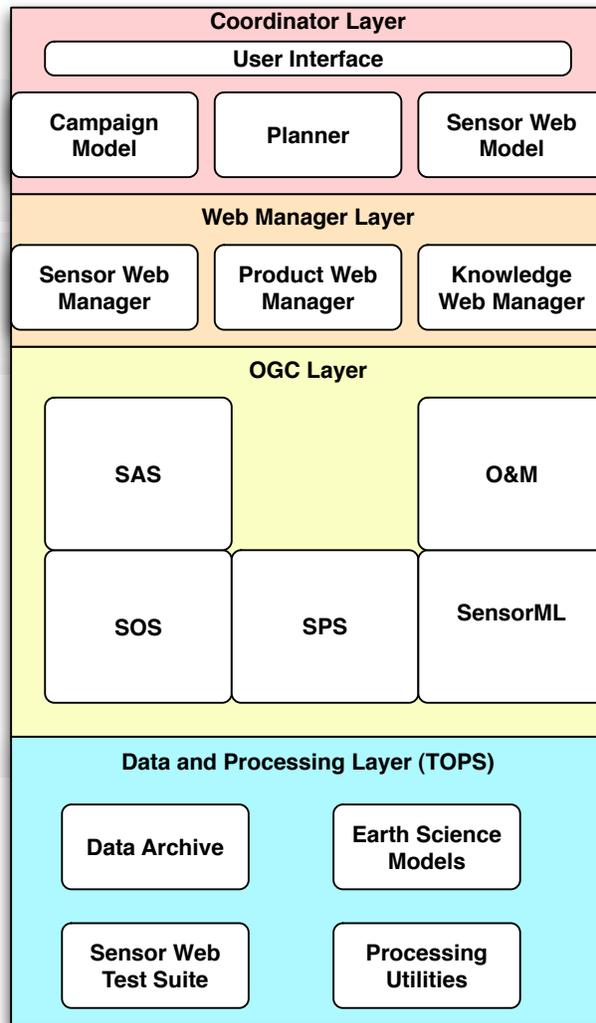


Figure 3: Software Architecture for Sensor Web Management

cution should adjust to unexpected behavior of the process, for example, by dynamically retargeting sensors. This is especially important in disaster monitoring applications where changes to the observing environment are happening relatively quickly and require a quick response by the observation system.

Robustness can be designed into an autonomous execution system in two ways: either by incorporating into the planning language constructs for *contingent execution* (in effect, allowing for branching sequences), or to enable *re-planning* during execution time. The GOES scenario offers a simple example of contingent execution. Future reports will focus on an investigation of the use of these forms of robustness into sensor web management.

```

(plexil-plan
  (list-node "n1_root"
    (variables (integer "frequency_of_goes" 15)
              (integer "last_goes" 0))
    (end-condition (> (lookup-on-change "localtime") 600))

  (list

    (list-node "n2_forecast"
      (variables (string "forecast_data"))
      (repeat-condition true)
      (list
        (sequence
          (command-node "Get_NDFD_forecast"
            (start-condition (> (lookup-on-change "forecast_no") 0))
            (command-with-return "get_forecast" (stringvar "forecast_data")))
          (if (= (stringvar "forecast_data") (stringval "clear"))
              (assignment-node "n3_setFreqClear"
                (assignment (intvar "frequency_of_goes") 240))
              (assignment-node "n4_setFreqCloud"
                (assignment (intvar "frequency_of_goes") 15))))))

    (while true
      (when (>= (lookup-on-change "localtime")
                (+ (intvar "last_goes") (intvar "frequency_of_goes")))
        (sequence
          (assign (intvar "last_goes") (lookup-now "localtime"))
          (command-node "Get_GOES_data"
            (variables (string "goes_image")))
          (command-with-return "acquire_goes_image" (stringvar "goes_image")))))
      )
  )
)

```

Figure 4: PLEXIL executable plan for GOES scenario in Lisp format