The COLLAGE/KHOROS Link: Planning for Image Processing Tasks

Amy L. Lansky
Mark Friedman
Lise Getoor
Scott Schmidler
Nick Short Jr.[**]

Recom Technologies/NASA Ames Research Center
MS 269-2, Moffett Field, CA 94035-1000

[**] NASA Goddard Space Flight Center, Code 935
Greenbelt, MD 20771

Contact: `LANSKY@PTOLEMY.ARC.NASA.GOV` (415) 604-4431

May 17, 1995

Abstract

This paper describes the application of the COLLAGE planner to the task of generating image processing plans for satellite remote sensing data. In particular, we focus on the linkage of COLLAGE to the KHOROS image processing system. Several obvious requirements presented themselves when we first confronted integrating COLLAGE and KHOROS: low-level connection tasks; representation translation tasks; the need to present users with a suitably coherent combined architecture. However, one overarching and pervasive issue became clear over time: how to represent and partition information in a way that fosters extensibility and flexibility. This is necessary for at least two reasons. First, KHOROS is an "open" system – its suite of image processing algorithms is constantly changing. Second, our combined architecture must be useable by a variety of users with different skill levels. These kinds of issues, of course, are common to many software engineering enterprises. Our experience with COLLAGE indicates that planning systems will also have to cope with them when they are used within operational environments.

1 Data Analysis Planning

The goal of this work is to apply domain independent planning methods to help scientists plan out their daily data analysis tasks. We are particularly interested in aiding Earth system scientists who study Earth's ecosystems using a mixture of remotely sensed data (satellite imagery) and ground-based data sets (e.g., vegetation studies, soil maps, etc.). Although these scientists are most interested in developing theories or models, they usually find themselves spending the bulk of their time puzzling over low-level data selection and manipulation tasks. Such tasks make up the "busy work" of their science. In the era of EOS (NASA's **E**arth **O**berving **S**ystem – a suite of satellites slated for launch in the next decade), scientists will have more data at their fingertips than ever before – an expected 1.2 terabytes/day. Fundamental innovations are required to keep the relatively small Earth science community from becoming swamped in the deluge.

For example, conducting even a relatively small study using one or two images can take weeks of a scientist's time. They may have to utilize two or three image processing or geographic information systems, each with its own set of algorithms, formatting requirements, and idiosyncracies regarding parameter usage. More often than not, each of these systems is resident on a different machine. To compound the problem further, a scientist must typically access several distinct databases to find the data they require.

Interestingly, similar problems are confronted by users of other types of software – e.g. graphic artists or users of other complex software tool kits. The heterogeneity and scope of such systems can create a logistical nightmare for their users. Although they may understand what they want to accomplish, users are awash in a sea of possible data, tools, and software routines. In the data analysis domain, scientists who can afford it hire technicians who specialize in data preparation. If they cannot, they muddle through, often using methods they are most familiar with rather than the ones that are most appropriate for their task.

In the context of software engineering and product design for such systems, there have been increasing efforts to create more integrated desk top environments to solve some of these problems. Indeed, the KHOROS image processing system we are working with is representative of one such effort [13]. Available for free over the Internet, KHOROS fosters an object-oriented approach to image processing. Users make use of and can augment a variety of toolboxes containing image processing algorithms. Algorithms can be selected from these toolboxes and combined to create visual image processing data flow diagrams (plans) using a GUI editor called **Cantata**. However, even with these tools, the expertise required to create such plans is substantial.

In the AI planning community, there have been growing efforts to automate parts of the data analysis process. For instance, several researchers in planning have begun to study how *data access* plans can be generated to aid users in finding the information they need [5, 16]. In contrast, our work focuses on aiding scientists in their use of the image processing and geographic information systems that sit on their desk. That is, given a high level task description, the goal of our application is to decompose it into a partially ordered set of steps corresponding to transformation algorithms executable on a particular platform. Other planning work in this vein is being done by Short [4, 15], Chien [3], Matwin [11], and Boddy [2].

The role of our planner can be viewed as much like that of an logistical assistant or technician [8]. While a data analysis planner does not require deep knowledge about a particular scientific discipline, it *can* be usefully embued with information about: the steps making up typical data processing tasks; the available algorithms on various platforms; and what the requirements of these algorithms are – their parameter settings, their applicability to various data types, etc. Interestingly, this is also the kind of mundane (yet volatile) information that a scientist would rather not deal with. The net effect is that the planner fills a role that is desired and valued, which also increases the likelihood of its eventual acceptance and use.

Of course, to be truly useful, a data analysis planner cannot sit in a vacuum. Ideally, it should be connected to the platforms on which the algorithms will be executed; the plan can be downloaded into the input format of a particular platform and executed there. A data analysis planner should also be connected to a framework that expedites data selection. Given an integrated planning architecture of this kind, a variety of issues must be reckoned with: utility (i.e. breadth and depth capability); ease of use; and openness to the natural evolution of component systems. The rest of this paper describes our experiences in building an architecture of this kind. Section 2 describes the overall framework and problems we have faced. Section 3 focuses on some of the larger issues that underly these problems.

2 COLLAGE/KHOROS Link: Core Issues

Figure 1 depicts the overall architecture of our integrated data analysis framework. In collaboration with a team from NASA's Goddard Space Flight Center, we are integrating the COLLAGE planner into Goddard's IIFS framework (the **I**ntelligent **I**nformation **F**usion **S**ystem) – an object oriented framework for ingesting and storing remotely sensed data, generating derived products and information about that data, and aiding the user in data selection [14] (see Figure 2). Scientists using the IIFS can utilize the KHOROS image processing system to create desired data products. COLLAGE serves as a front-end to this framework to plan out exactly what KHOROS algorithms should be used to achieve a particular task.

Ultimately, it is also our intention to link other image processing and geographic information systems (GIS) into this framework. A GIS deals with map data in addition to image data, and in particular, enables correlation between these two types of data. The most likely GIS choice for incorporation into our framework is Arc/Info [1]. Besides our collaboration with NASA Goddard and the IIFS, we also plan to link Collage/Khoros into another data access framework being built at the Lockheed AI Laboratory [16].

Collage itself is a nontraditional domain-independent planner with several unique features [6, 7, 8, 9]. Of these features, the most relevant to this application are Collage's domain description language and planning methods, which are based on the use of *action-based constraints*. This approach contrasts rather sharply with the traditional planning representation and methodology that utilize STRIPS-based (i.e. state-based) reasoning [7]. In particular, all planning requirements are specified in terms of required forms of action instantiation, action-decomposition, temporal and causal relationships between actions, and "CSP-based" [10] binding requirements among action parameter variables. No explicit preconditions or goal states are used. For example, in the data analysis domain, planning requirements are specified by, first, requiring the addition of high-level "task" actions into the plan (via Collage's `action` constraint form). These high-level actions are then incrementally decomposed (via Collage's `decompose` constraint form) into lower level actions, and ultimately into actions which represent specific algorithm instantiations.

One important feature of Collage's method of constraint satisfaction is that any constraint can be conditionalized upon the emerging form of the plan as well as information in a static *knowledge base* that incorporates domain-specific facts and functions. For example, depending on the form of the plan or scientific domain information, a high level action might be decomposed in one of several possible ways. The domain knowledge base facts and functions can also be used to functionally define the various binding requirements that are imposed on plan variables during the constraint satisfaction process.

Figure 3 depicts the overall Collage architecture. Besides a constraint-based approach to planning, this figure also shows another unique aspect of Collage – its use of *localized* or partitioned reasoning spaces. In particular, the set of constraints that make up a problem/domain description may be partitioned into sets called *regions*. Each regional reasoning space is focused on creating a regional plan that satisfies regional constraints. Regions may be interrelated in fairly arbitrary ways; e.g. they may form hierarchies or share subregions. The job of Collage's localized search mechanism is to make sure that the overall "global" plan is consistent and satisfies all regional requirements. In general, the planning cost savings provided by localized search can help deal with the problem of scaling up to large domains [6]. In this application, however, where scale is less of an issue, localization provides a useful mechanism for structuring domain constraint information.

Once a Collage plan has been generated for a particular data analysis task, it can be automatically translated for use by Khoros. This is done via a two-step process. First, the plan is translated into an intermediate form, similar to a data flow graph. This form is then translated a **Cantata** workspace file. **Cantata** is the visual programming environment for Khoros. The workspace format serves as the storage representation for **Cantata**'s visual programs. After a Collage-generated workspace file is loaded into **Cantata**, it can be directly modified or manipulated there and executed by Khoros. Figure 5 depicts Khoros's **Cantata** visual programming environment, loaded with a Collage-generated plan, a fragment of which is depicted in Figure 4. This plan derives a vegetation index that quantifies biomass from images taken by the NOAA AVHRR instrument. The visual inset on the bottom-right shows the result of executing this plan on an image of the Pacific northwest. The more lightly colored areas depict areas of high concentrations of biomass. The inset on the lower-left shows an interaction menu for one of the plan steps and illustrates **Cantata**'s interface for modifying algorithmic parameters.

When we originally faced the task of linking Collage to Khoros (and the IIFS), we easily recognized several tasks that had to be accomplished:

- *Creating a low-level link between systems.*

In a first crack at this problem, we have considered only the transfer of information from Collage to Khoros (rather than vice versa). This was achieved via translation to the workspace format. Ultimately, we will also consider how to send information back from **Cantata** to Collage, in

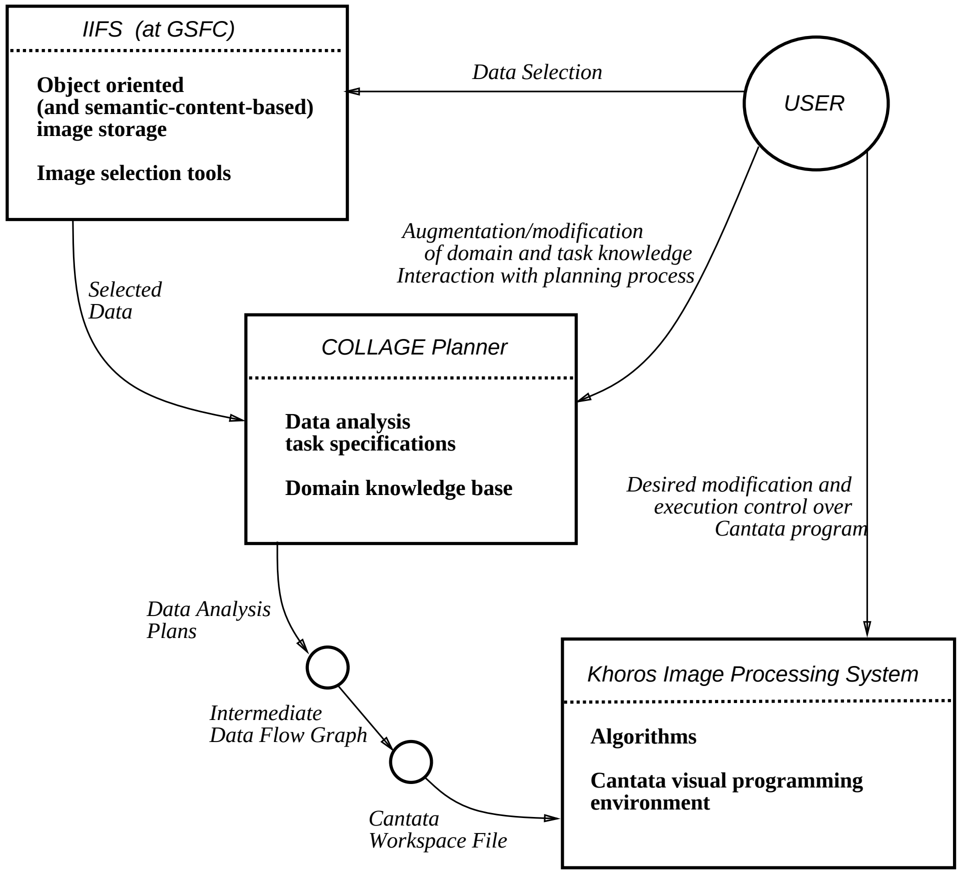

Figure 1: Integrated Planning Architecture for Data Analysis

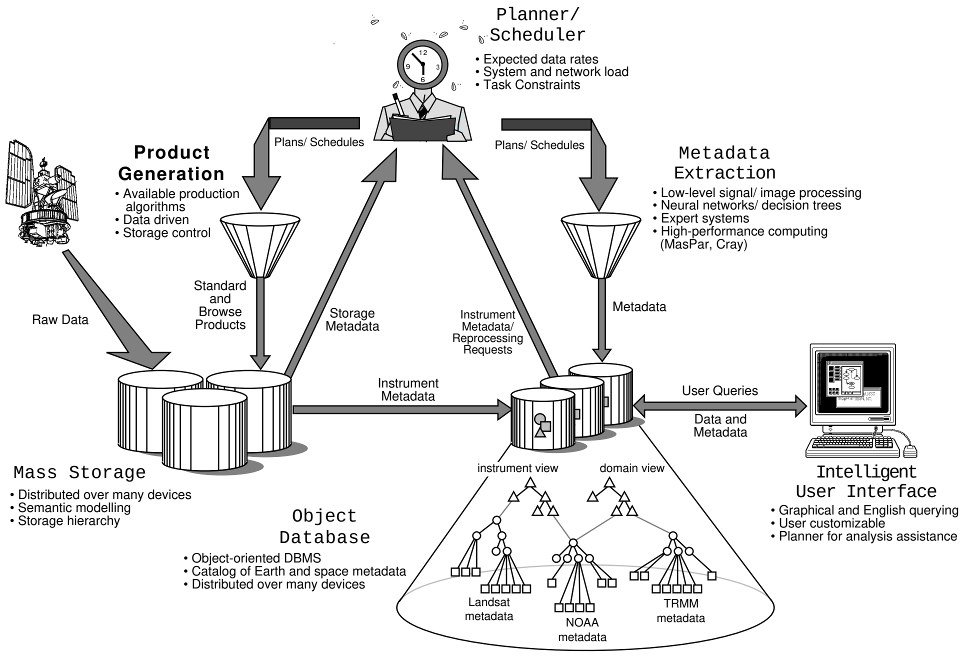

Figure 2: Intelligent Information Fusion System Architecture

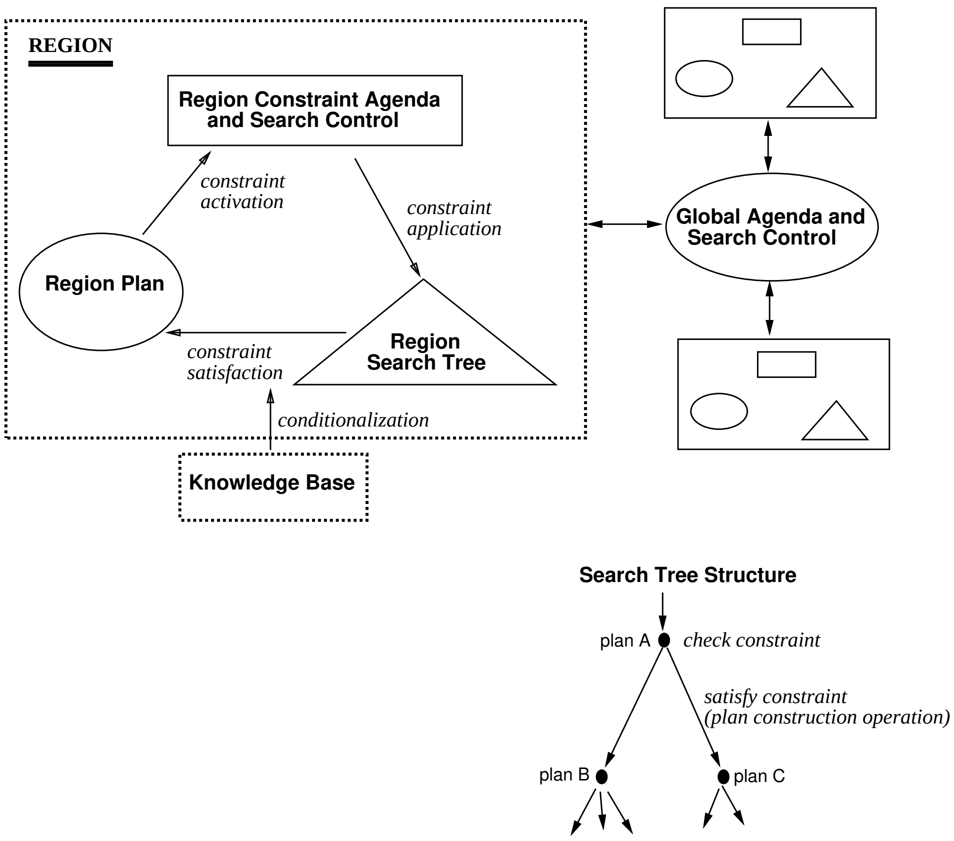

Figure 3: COLLAGE Architecture

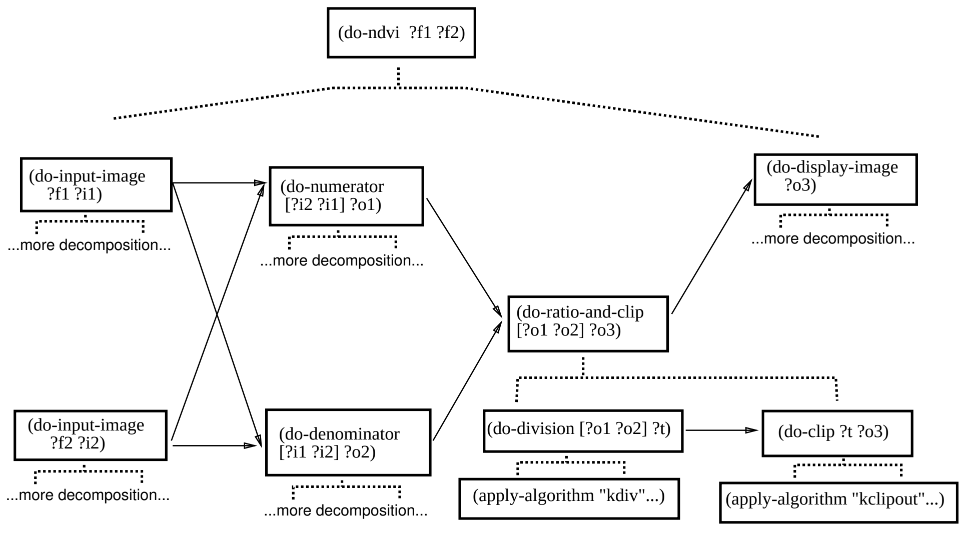

Figure 4: A simple plan fragment

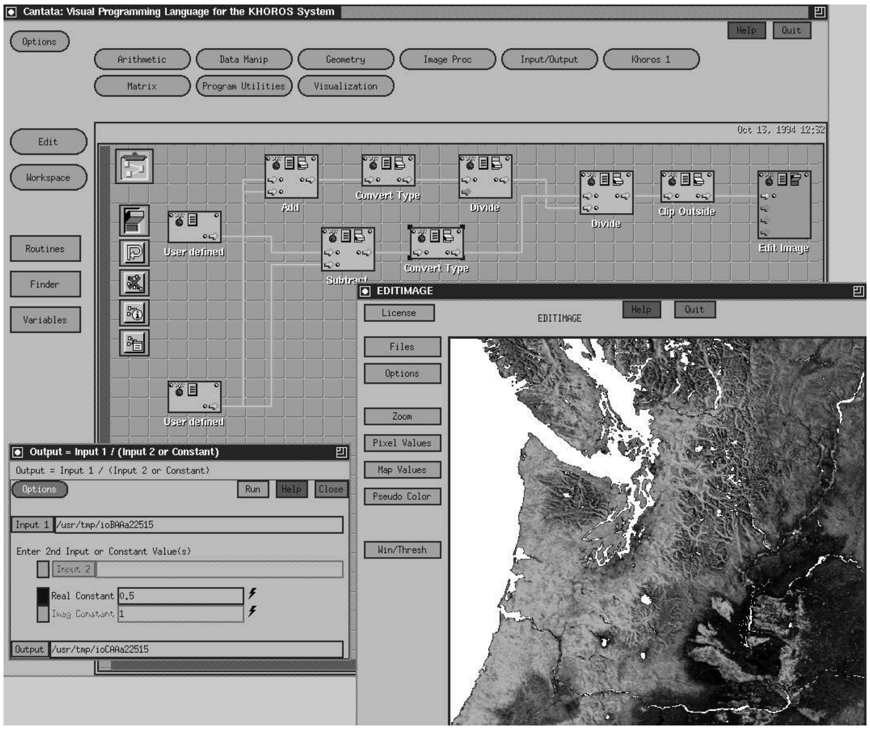

Figure 5: COLLAGE plan utilized in Khoros/**Cantata** environment

service of more reactive planning capabilities (e.g. reactions to plan-modifications made by the user using **Cantata**). This will probably be achieved via a reverse translation process that decodes workspace information into a form usable by Collage.

As far as the translation process itself, this task required us to do reverse engineering on the workspace format, since it isn't documented nor meant for user viewing and editing. We also designed the intermediate data-flow-graph as a structure that was amenable to translation into the workspace format. Our hope is that this intermediate form will also ultimately serve as an "interlingua" between Collage and other image processing and GIS systems. Finally, we also had to write our own mechanism for automatic graphical layout of the plan components, since this layout is normally provided manually by the **Cantata** user as they build their data flow plans.

• *Deciding upon a desired form for* Collage-*generated plans that is amenable to translation.*

A natural corollary to this task was figuring out how to write domain specifications that could generate such plans – i.e. we had to come up with a "template" for writing task specifications for the data analysis domain.

The most problematic feature of this task was representing the parameters of the image processing algorithms and the data flow relationships between these parameters. Since the suite of available Khoros algorithms is constantly in flux, we also wanted to have a single generic action-type that could be instantiated to represent the application of any type of algorithm, rather than a separate type for each algorithm. This action-type description is given below.

```
:action-type
  (apply-algorithm
    ?n_algname ?i_inputparams
    ?o_outputparams ?p_otherparams)
```

Notice how this representation identifies the algorithm name `?n` as a parameter. It also utilizes two distinguished input and output parameters `?i` and `?o`, which are bound to lists of input and output parameter variables that are used by the algorithm. In particular, the input parameters are received from other algorithms and the output parameters are sent to other algorithms – i.e. these parameters play a role in the data-flow diagram created for **Cantata**. The final parameter `?p` is another list of variables (algorithm parameters) that are also required by the algorithm but are not linked to other algorithms. To represent the data flow relationship between specific algorithm parameters, we used a combination of "CSP"-style relations between relevant variables (in particular, between `?i` and `?o` subvariables) and a special `pipe` relation between the `apply-algorithm` action instances.

Notice that while most planners only allow for action parameters that can be bound to simple atomic values, our representational choice for `apply-algorithm` required us to allow for action parameters that could be bound to indefinite lists of subvariables (i.e. lists of actual parameters passed for a particular algorithm). Indeed, because of the complexity of parameter information in this domain, we also needed to allow for other forms of structured parameter variables. Coupled with this was the required ability to impose and propagate binding requirements on these variables and their component parts.

For example, many of the parameters of our COLLAGE actions (as well as those ultimately passed down to KHOROS algorithms) are pointers to images, each associated with many features (e.g., time, location, and formatting information). Since the planning process requires reasoning about these features, it is most suitable to represent these "image" variables as a records of information, whose slots are filled by other variables. All of these variables and subvariables must be accessible to the binding propagation facility. We recognized these needs early on in this effort, and focussed several months of project time augmenting COLLAGE with these capabilities. The foreseen requirements of this domain were also the driving force behind our incorporation of a static domain knowledge base and planning capabilities that conditionalize the planning process off that knowledge.

• *Presenting users with a coherent view of a heterogenous system.*

This problem is complicated by the fact that we forsee many different kinds of users for this framework, with different skill levels. For example the bulk of the users will be scientists who will only be interested in a very high-level view of both COLLAGE and the IIFS (basically, for selecting data and posing task goals); most of their interactions will be directly with KHOROS. At the other end of the spectrum are the COLLAGE and IIFS developers. In the middle are computationally sophisticated scientists who may develop new algorithms for incorporation into KHOROS as well as new task specifications for COLLAGE.

Our first crack at this problem was to utilize the same user-interface framework for both COLLAGE and the IIFS; we are currently developing TK-based [12] front ends for both systems (see Figure 6). We also plan to make the mode of interaction with TK similar to one that is utilized heavily in KHOROS – "forms-based" interaction. A scientist end-user will thus interact with **Cantata** and high-level TK-based views of COLLAGE and the IIFS in much the same way. More sophisticated users will interact at a deeper level with COLLAGE, but also via TK. We are currently working on a specification builder for COLLAGE that will utilize a TK-based GUI interface. This will replace the current mode of creating Lisp-based task specifications using a text editor. It will foster flexible structuring and configuration of specification information and enable sharing and reuse among various task applications.

Our most intensive work on this data analysis framework has been going on since the spring of 1994. Since then, we have found many of the tasks described above to be solvable. However, one basic issue has emerged as persistent and problematic: the inherent "openness" of the environment we are targetting. Thus, COLLAGE is facing problems common to most large, open software development environments.

For example, in the past six months, we have already reckoned with two new major releases of KHOROS. Each new release manifested a complete reorganization and, in one instance, a complete overhaul of the KHOROS algorithm suite. Even the workspace format changed. It is clear that our architecture must take into account the fact that KHOROS (and indeed any image processing or GIS system we link into) will remain a moving target. The same type of problem arises in the development of our constraint task specifications; i.e. the suite of tasks we wish to cover will always be changing or restructuring. We soon recognized the need to write and structure our task specifications so that certain portions could be shared and so that necessary changes (e.g. to accommodate algorithm modifications within KHOROS) didn't propagate wildly throughout the specification code. As in other software enterprises, we have found that the key to

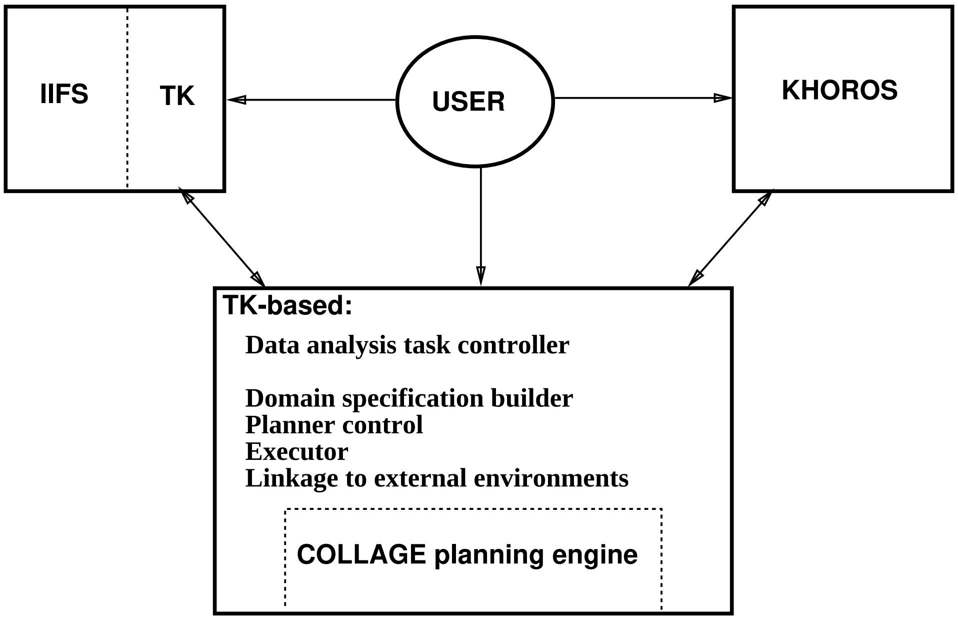

Figure 6: Creating a coherent user interaction environment

openness, reusability, flexibility, and modifiability is an appropriate use of structuring techniques – i.e., the partitioning and abstraction of information in appropriate ways.

3 Planning in open environments

One way to view the "openness" problem and many of the other tasks described in the previous section is in terms of representation and language issues. There are several different "languages" at work in our framework: COLLAGE's domain specification language; COLLAGE's plan language or representation (i.e. the form of generated plans); the intermediate data flow graph language; **Cantata**'s workspace format; and the user-interface "languages" presented by all three systems. Most of the discussion in the previous section dealt with language translation or expressiveness. System openness must be handled by focussing on language structuring, information hiding and layering, and reconfigurability.

Figure 7 depicts the ways we have handled these problems thus far. Quite early on in this project we recognized the need for a knowledge base of facts and functions distinct from COLLAGE's constraint-based task specifications. The knowledge base is used to conditionalize the application of constraints and to define functions used by variable binding requirements. By keeping the knowledge base distinct, the constraints used in the task specifications can be reused in different knowledge-base contexts. This obviously fosters reusablity.

However, it also serves the function of keeping different kinds of information and information of interest to different communities physically distinct. For example, information about Earth-science-specific data types, facts, and functions is understandable to scientist users and should be open to and modifiable by a broad range of those users. In contrast, the COLLAGE constraint specifications that specify how various tasks should be decomposed are of interest to a much smaller community. There are only a limited number of data-analysis task forms; once defined, these can be reused by all system users. And since learning the COLLAGE constraint language is more difficult than modifying some predicate-based facts in the knowledge base, we felt it was best to shield most scientists from this information by focussing most of their interaction on the knowledge base.

Of course, even within the knowledge base, there is useful partitioning of information. For example, it contains information about the available KHOROS algorithms as well as information about specific im-

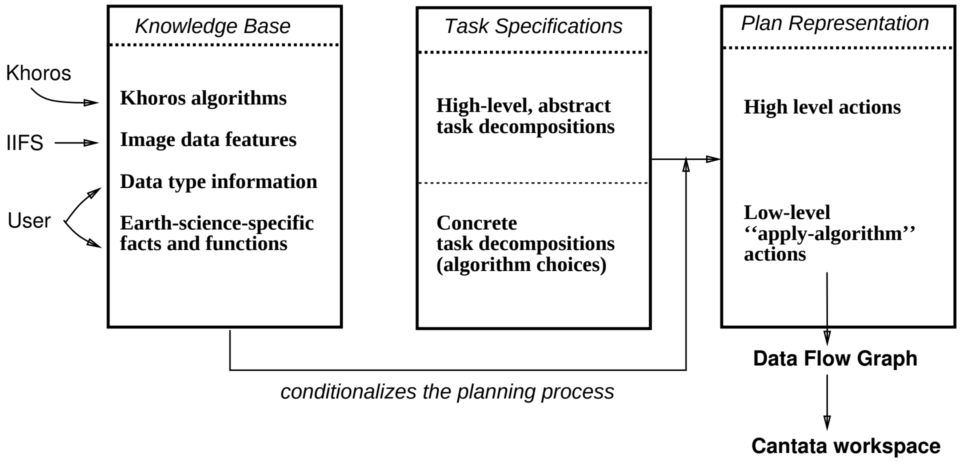

Figure 7: Structuring of information

ages obtained from the IIFS. Both kinds of information could profitably be downloaded automatically, which would in turn greatly enhance the overall system's ability to remain open to changes within KHOROS or the IIFS.

There are also natural levels of abstraction or partitioning apparent within the suite of constraints that make up the task specifications. These levels are also reflected within the plans themselves. During the planning process, the highest level task actions are decomposed into more detailed high-level task descriptions without any reference to specific algorithm choices. Indeed, there may be multiple levels of decomposition at work within this more abstract context. At the bottom level, however, the lowest level task descriptions bottom out in the choice of an algorithm that performs that task (see Figures 4 and 7).

Our task specifications directly reflect this partitioning into levels via use of COLLAGE's region mechanism. We have found that this partitioning has numerous advantages. The most important is that only the lowest level constraints need be modified if the algorithms within KHOROS change. These low-level constraints can then be *reused* or *shared* between different high-level task contexts, which greatly alleviates the specification task. It also enables a broader range of users to write task specifications; scientists can easily write high-level task specifications while ignoring low-level platform-specific detail. The issues of flexibility, configurability, and information-hiding have also been the driving force behind our development of our new TK-based interface framework for COLLAGE. This framework will foster incremental development, reuse, and flexible configuration of the domain knowledge that COLLAGE utilizes, as well as linkages to the other systems.

4 Conclusion

This paper described the connection of the COLLAGE planner to the KHOROS image processing system and NASA Goddard's IIFS data framework. The initial problems we faced in this application included making low-level connections between the systems, expanding the expressivity of our planning language, and a host of translation tasks. The ongoing issues that now confront us deal more with the inherent openness and volatility of this framework. Our solution thus far has been to focus on underlying representation issues; i.e., how system components and information should be compartmentalized, shared, expanded, and reconfigured. In the future, we expect that much more work will be needed to foster domain knowledge capture and to ensure user accessbility to this complex heterogenous system.

References

[1] *ARC/INFO User's Guide*, Environmental Systems Research Institute, 380 New York Street, Redlands, California 92373.

[2] Boddy, M., White, J., Goldman, R., and Short, N., "Planning Applications in Image Analysis", 1994 NASA-GSFC Proceedings of Space Applications of AI, Greenbelt, MD, May 1994, pges 17 - 28.

[3] Chien, S. "Using AI Planning Techniques to Automatically Generate Image Processing Procedures: A Preliminary Report," *Proceedings of the Second International Conference on AI Planning Systems*, Chicago, Illinois pp. 219-224 (1994).

[4] Cromp, R. F. and Campbell, W. J. and Short, Jr., N. M., "An intelligent information fusion system for handling the archiving and querying of terabyte–sized spatial databases," *International Space Year Conference on Earth and Space Science Information Systems*, American Institute of Physics, 1992.

[5] Knoblock, C. and Y. Arens, "Cooperating Agents for Information Retrieval," in *Proceedings of the Second International Conference on Cooperative Information Systems*, University of Toronto Press (1994).

[6] Lansky, A. "Localized Planning with Diverse Plan Construction Methods," NASA Ames Research Center, Artificial Intelligence Research Branch, Technical Report FIA-94-05 (1994). Under review for AIJ.

[7] Lansky, A. "Action-Based Planning" in *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS-94)*, Chicago, Illinois, pp. 110 -115 (1994).

[8] Lansky, A. "A Data Analysis Assistant" in *Proceedings of the 1994 AAAI Stanford Spring Symposium on Software Agents*, Stanford University (1994). Also in *Proceedings of the Third International Symposium on Artificial Intelligence, Robotics, and Automation for Space (I-SAIRAS 94)*, Pasadena, California, pp. 373-377 (1994).

[9] Lansky, A. and A. Philpot, "AI-Based Planning for Data Analysis Tasks," Proceedings of the Ninth Conference on Artificial Intelligence for Applications (CAIA-93), Orlando, Florida, pp. 390–398 (March 1993). Also appeared in *IEEE Expert Magazine*, Volume 9, Number 1, February 1994.

[10] Mackworth, A. K. "Consistency in Networks of Relations," *Artificial Intelligence*, Volume 8, pp. 99-118 (1977).

[11] Matwin, S., Charlebois, D., and Goodenough, D., "Machine Learning and Planning for Data Management in Forestry," Workshop on AI Technologies for Environmental Applications, 1994 National Conference on Artificial Intelligence, July 31 - August 4, 1994, Seattle, WA, pges. 83 - 90.

[12] Ousterhout, J.K. *Tcl and the Tk Toolkit*, Addison Wesley Publishers (1994).

[13] Rasure, J.R. and C.S. Williams, "An Integrated Data Flow Visual Language and Software Development Environment," *Journal of Visual Languages and Computing*, Volume 2, pp. 217–246 (1991).

[14] Short, N. et. al., "AI Challenges within NASA's Mission to Planet Earth," Workshop on AI Technologies for Environmental Applications, 1994 National Conference on Artificial Intelligence, July 31 - August 4, 1994, Seattle, WA, pges. 1 - 15.

[15] Short, N. and Wattawa, S. L., "The Second Generation Intelligent User Interface for Crustal Dynamics Information System", *Telematics and Informatics*, Volume 5(3), pges. 253 - 268.

[16] Toomey, C.N., Simoudis, E., Johnson, R.W., and W.S. Mark, "Software Agents for the Dissemination of Remote Terrestrial Sensing Data," in *Proceedings of the Third International Symposium on Artificial Intelligence, Robotics, and Automation for Space (I-SAIRAS 94)*, Pasadena, California, pp. 19 - 22 (1994).