

# Negotiation in Network Based Requirements Analysis

Balasubramaniam Ramesh  
 Georgia State University  
 Atlanta, GA 30303, U.S.A  
[Bramesh@gsu.edu](mailto:Bramesh@gsu.edu)

Tung Bui  
 University of Hawaii  
 Honolulu, HI 96822  
[tbui@busadm.cba.hawaii.edu](mailto:tbui@busadm.cba.hawaii.edu)

## Abstract

*For each of the stakeholders involved in large-scale, complex and distributed software development processes, some useful support can be provided by recording the argumentation among team members. We describe a system that supports argumentation based on the ARBAS language. A variety of support mechanisms for the capture and use of software development process knowledge are discussed. These include support for the capture, retrieval, and replay of process knowledge as well as mechanisms to maintain the consistency of it.*

## 1. A process View of Supporting Collaboration in Software Development

The development of systems to support collaboration among system development team members has received much attention in recent research. However, systems that attempt to provide analytical decision support to groups, such as those that employ decision theoretic models assume that the alternatives to solve the decision problem of interest are well understood and the effects or consequences of choosing one or more alternatives are well known. The process of selecting from among alternatives involves using utility functions that take into account the goals in the decision-making context. In group decision-making situations, the identification and elaboration of goals (or selection criteria) and alternatives that resolve the issues is very critical for arriving at a consensus or group solution. This often leads to controversies and deliberations to resolve them. We believe that process of the elaboration and refinement of the issues, alternatives and decision criteria itself is an important component of the problem solving and may guide a group towards a solution. Finally, support for facilitating group communication can be most effective if it is based on a process model. Another important component of knowledge about group decision making process is the knowledge about arguments behind the selection of decision criteria and alternatives. In, analytic/decision theoretic approaches, the

knowledge about "rationale" often get implicitly reflected in the payoff functions associated with alternatives. Furthermore, the assumptions that are embedded in the choice of alternatives and decision criteria are not explicitly captured. This information is essential to ascertain the validity of a decision in a changed context characterized by changed assumptions. Systems that aim to support communication focus on supporting information structuring and exchange rather than on reasoning with the information. Their emphasis is on capture and communication of information, rather than providing intelligent decision support with the captured and exchanged information.

Our research addresses these issues by providing a framework and mechanisms to facilitate information exchange and communication among group members based on a model of the group decision making process. Our group decision support system provides a mechanism for conducting deliberations on the actions and their impact on resources and maintains this knowledge in a context characterized by changing assumptions.

## The ARBAS Language

The benefits of supporting semi-structured communication among group members have been well demonstrated by recent research. As a first step in providing such a support, we have developed tools to support a model that characterizes the communication among group members. In the domain of information systems development group problem solving is very critical. Systems design in the large is characterized by project teams with thin distribution of knowledge among members. Various participants in the design process bring in different viewpoints and expertise towards developing a solution. We view conflicts as a part of the deliberation process through which team members share knowledge. We have developed an argumentation language called ARBAS to represent negotiation among groups involved in such deliberations.

The ARBAS language is described in detail elsewhere [1]. Figure 1 illustrates some of the important components of

an ARBAS argumentation and can be briefly explained as follows: In ARBAS, argumentation is focussed on actions and their impact on resources. A decision problem involves the identification of the actions to be taken and the resources that are affected by these actions. An action may require input resources and/or generate output resources.

Negotiations are structured using views on actions that may represent sentiment, opinion, belief, conviction and persuasion. The participants in a negotiation can substantiate their views by relating them to relevant actions. Justifications in the form of free form text (or formal declarations when feasible) are provided for the various views. The participants may also propose various possible moves (e.g., drop, implement, modify) that affect the activity. ARBAS supports problem evolution by a chain of activity triplets (consisting of input resources, action and output resources) over time.

Analogous to the Data Manipulation Language in relational DBMS, ARBAS provides an argumentation language that helps users search for information according to discussion topics, position of a party on a particular issue, evolution of arguments or viewpoint overtime, etc.

The interaction between initial goals and design possibilities are brought out by communication among group members. This process facilitates incremental development of a solution guided by evolving the evolving requirements. In our model, an argument leading to the consideration of various alternatives can trigger another discussion to consider criteria for the evaluation. The process of selection of criteria is also viewed as an argumentation process in our model.

Stefik et al. [12] postulate that much of the dispute and misunderstandings in group meetings about design proposals are due to three major causes: owned positions, unstated assumptions and unstated criteria. Our approach provides a framework that mitigates these problems as follows:

*Owned positions:* Each participant may favor a position based on personal preferences. In ARBAS, each participant is expected to clearly state the arguments for or against their positions. Justifications that are not based on objective criteria or shared beliefs and preferences can then be easily identified.

*Unstated assumptions:* There may be hidden assumptions behind the positions taken by participants. As other group members who do not understand these assumptions, such a situation can lead to conflicts. In our approach, the assumptions behind arguments have to be explicitly stated. The other participants may then be able to understand the

conditions under which an argument is valid.

*Unstated criteria:* The criteria for choosing among alternatives are often not stated explicitly. The participants may be evaluating alternatives based on differing criteria. This could lead to conflicting choices. Furthermore, the participants may even be unaware that the others may be working with a different set of criteria. In our model, a discussion about the criteria for evaluation of alternatives could be posted, and various views discussed. Thus arriving at a group norm for criteria to be used in evaluating alternatives is facilitated.

## 2. Teamwork and Traceability

Recent literature emphasizes the importance of mechanisms to capture and manage rationale in situations involving large number of participants involved in complex organizational processes like large-scale systems development [3]. In collaborative activities involving a large number of participants, each having a different set of goals and priorities, maintaining a comprehensive history can be invaluable.

A variety of stakeholders involved in organizational decision processes bring together their often unique viewpoints and expertise. These range from top management providing inputs on the nature of the problem and its impact on the organizational goals, to middle managers who help articulate objectives and constraints, to lower level workers that implement them. Further, these situations are also characterized by incomplete and even inaccurate information. In such complex decision situations, a comprehensive history of the context in which decisions are arrived at and the ability to understand and analyze decision implications when they change will be extremely helpful. The need to capture and reason with the information regarding the *context* is even more pronounced when the decision making includes participants with different objectives. Silverman [11] emphasizes the importance of facilities to simulate, structure and map alternative views of the problems. The decision making process can then be viewed as a reflective learning process that involves the sharing of perspectives and viewpoints. Recent studies confirm that capturing organizational memory is especially important in large-complex activities for the following reasons:

- The context in which key decisions are made will be lost when different teams are engaged in various aspects of decision making [9].
- Critical errors are commonly made in the formulation

- and resolution of decisions, but they are often unnoticed
- in the absence of comprehensive traceability to the context in which they are made [14]
- In the absence of reliable traceability, work groups often engage in repetitive discussions of the same issue [14].
- When stakeholders with different perspectives and viewpoints are involved in activities such as software development, key decisions are often misunderstood.
- and misinterpreted [3].

Recent studies argue comprehensive traceability to decision processes is most useful when it increases the shared understanding of the users about an organization's history. Complex decision making situations are characterized by changing environments and are subject to incomplete information, misinformation, uncertainty, and changing preferences. Each of these characteristics of complex decision making situations supports the need for a comprehensive organizational memory of the context in which decisions are arrived at and the ability to *understand and analyze* its implications when it changes. The need to capture and reason with the information about the "context" is even more pronounced in complex decision making situations that involve the use various "functional" domains. Such traceability provides the ability to modify and manipulate decisions in response to changing requirements and assumptions that are valuable in diagnostic problem solving. Silverman [11] supports this view emphasizing the importance of facilities to "simulate, structure and map alternative views of the problems to view decision making as a reflective learning process".

As discussed in the last section, a language such as ARBAS is necessary for representing argumentation in system design activities. We now discuss the capabilities of a prototype system that facilitates the acquisition, maintenance and use of argumentation knowledge and automated tools for supporting the needs of various stakeholders are discussed. This is followed by a discussion on related work and plans for future work.

### 3. Negotiation Support Environment

We have developed a prototype environment to support the negotiation process using the primitives of ARBAS.

We describe the functionalities of this environment using a scenario in systems development. Various facilities for supporting group deliberations, maintenance of dependencies among components of deliberations, reasoning with temporal information, supporting ad-hoc retrieval of argumentation knowledge and replaying information about argumentation

are discussed.

Our prototype environment supporting the capture and reuse of argumentation knowledge has been built using ConceptBase [4], an implementation of the high level conceptual modeling language Telos [7]. The system is based on a client-server architecture so that negotiations among group members distributed across a wide area network can be supported. This environment provides deductive object management system capabilities to reason with captured knowledge. Deductive rules can be used to make inferences as negotiations proceed. Constraints can be used to maintain the integrity of the knowledge base. A reason maintenance system maintains the dependencies among various knowledge components. The system provides a graphical user interface. Access to contextual information is made possible using interfaces to the World Wide Web.

#### 3.1 Support for Group Deliberation

Our environment supports the capture of argumentation knowledge by providing a mechanism for teams to conduct their deliberations using the primitives of our model. As the deliberation proceeds, various actions that need to be taken, the views that respond to them, justifications behind them are defined by the users. A hypertext like view of the knowledge base helps visualize the stages in the argumentation process. Associated with each instance is a distinct menu of choices that is defined for its class. These menu choices, in effect, define the legal moves possible during the deliberation process. Our system also provides a textual view of the knowledge base with a frame-based representation.

Figure 2 shows the results of a deliberation session among a set of system designers. The deliberations involve a series of actions that need to be taken in coming up with a design for an order entry system. In the figure, instances of different primitives in ARBAS are displayed as icons with different shapes, and relationships are displayed as links. The classes to which the nodes belong are also shown on the left.

A design team engaged in the deliberation on the choice of a user interface for a transaction processing system. The issue being discussed concerns the various items that need to be included in the display of an order processing system when an order is taken. The stakeholders involved in the discussion are the finance department, the sales department, and the IS department. The IS department is interested in meeting the requirements of the other two departments subject to the constraints schedule and costs. Initially a design option (DESIGN1) is proposed by the finance team. It is

based on the view that a field to show the additional field showing inventory costs needs to be added to the design. This view supports the action to implement DESIGN 1. Finance department also provides the justification for this request stating that inventory costs are crucial in the organizations effort to reduce costs. At this stage in the negotiation process, the provider and the consumer of resources for the action correspond to the IS department (providing labor) and the Finance department (receiving additional functionality) respectively.

The sales department also joins in the discussion at this stage. Figure 1 shows that a new design (DESIGN 2 derived from DESIGN 1) is proposed by this department. This design includes two additional fields: location of the clients and the type of packaging boxes. At this stage, the IS department remains the provider of the resources, but the Sales department in addition to the finance department is the on product packaging are necessary to sustain their market segmentation consumer of resources. The sales department presents its conviction supporting this design and requests its implementation. This view is based on the justification that information on the customer's location and detailed information strategy.

At this stage in the negotiation, it is obvious to the IS department that it can not meet these increased functionalities within constraints on resources that they need to operate with. They present their view that there are inadequate resources (e.g., staff, schedule) available to meet the demands of these two departments. Therefore, they suggest that DESIGN2 be dropped from further consideration.

The management gets involved in the review of priorities and resource allocation. As suggested by them, the advertising department is willing to lend the services of a graphics designer to help with the user interface development. As even with this additional resource, DESIGN2 does not appear to be a viable option. Sales department then proposes DESIGN3 as an alternative. This design includes only one rather than two elements to be included in the order entry screen, with the understanding that the other will be included at a later time. DESIGN3 is considered an acceptable solution by all the concerned parties and is accepted as the action for final implementation.

The above scenario illustrates how the various team members can interactively define and modify components of their arguments. Each participant proposes different categories of views and provides corresponding justifications. As these are explicitly represented, the assumptions and viewpoints of each of the team members become readily

available for review by other members, leading a consensus.

### 3. 2 Support for Dependency Maintenance

Our model explicitly identifies the dependencies among its primitives. Our decision support environment provides mechanisms for maintaining and reasoning with these dependencies with a special purpose Reason Maintenance System.

In our model, the belief status of a view will depend on the belief status of justifications that support it. A dependency network can be specified using deductive rules defined over instances or object classes. For instance, a deductive rule may be specified to ensure that the belief in a view is the same as the belief in the justification that it depends on. It is specified inTelos as follows:

```

Individual View with
Rule
  Beliefrule: $
  Forall v/view
  Forall j/justification
  ((j belief in) and
   (j supports v)
   → (v belief in)) $
END
    
```

The above rule provides a way for automatically inferring the belief values of various views expressed by different stakeholders and helps in identifying valid views, i.e., those that are supported by current set of beliefs in applicable justifications. For example, in Figure 2, the view from the IS department to drop DESIGN2 is based on the justification that there are no adequate resources available. The management may take resource reallocation decisions (such as authorizing new hires, reassignment of IS personnel from other projects to the current one) that may invalidate this assumption. This can be easily done in our system by choosing a menu of choices available on the justification. Now, using the deductive rule, the system will automatically identify DESIGN2 as a valid option as the view that suggests that it be dropped from consideration has lost its support. Nodes that have no valid support in our system may be shown with different colors to visually identify candidates for reevaluation.

Though we have chosen a simple scenario to illustrate the use of dependency management, this facility can be used to manage dependencies among an arbitrarily large network of actions and related components. In large projects, that are

characterized by some complex network of interdependent actions, the ability to automatically identify the valid actions with changing requirements and assumptions will be extremely valuable. The lack of such ability is a primary reason inconsistent decision taken by different groups working on different aspects of a large project. In our example, another topic for group decision may be the design of inventory audit reports. This decision will be directly affected by the decision on the data items to be included in the order processing input screen. For instance, the decision to leave out inventory cost data may exclude from consideration several reports that require this data. In our system, this information can be specified as an integrity constraint. The integrity constraint can be thought of as specifying a contradiction in reason maintenance system. Any assertions into the knowledge base that lead (directly or indirectly) to the violation of such a constraint will be rejected by the system.

### 3.3 Temporal Reasoning

Another area of decision support provided in our system is temporal reasoning. Validity time, which is the duration during which an object or attribute is valid can be specified by the users. For instance, temporal information such as the time interval during which a justification qualifies a view can be captured. This facility can help trigger automatic computation of properly justified views at any specified time period. In our example, if the justification about the lack of resources in the IS department may be declared valid only for the next two months (based on plans for new recruiting efforts). This can be specified in Telos as follows:

```
Individual inadequatestaff with
Attribute
Supports: justifies at
        <time-relation> <time-interval>
END
```

Here the time-relation could denote an open or definite time interval such as before, after, during, equals etc. Time-interval can be user defined or may be one of the predefined such as year, day, month, minute etc.

At the end of this time period, the justification will be automatically declared invalid by the system and hence, DESIGN2 will become a viable option. Then, the group decision about DESIGN2 may be re-evaluated using the current set of choices.

An important feature of ARBAS is the representation of the dynamics of activities (or decision processes) during the course of the negotiation by chaining activity triplets over

time. As activities get modified and refined over the negotiation process, facilities to reason with the temporal information about activities and fragments of arguments about them is very valuable. However, it should be noted that temporal reasoning is computationally very expensive and needs to be specifically enabled in our current implementation.

### 3.4 Support for Retrieval

Support for various stakeholders can be provided by constructing ad-hoc deductive queries provided by our system. Queries are defined as special classes whose instances are answers to the queries. For example, the answer to a query can be used by a senior manager to identify the ramifications of various critical assumptions that are part of the justifications behind various views put forward by different participants. The queries can be defined and the answers displayed in a graphical browser or in a frame-based representation. The query language supports sophisticated retrieval, including recursive queries.

The following definition in Telos can be used to retrieve all the actions that are supported by valid views (i.e. those views that have valid justifications behind them).

```
Queryclass valid_actions isA view with
Constraint
Valid_action:
    $ exists j/valid_justification
                v/views
    (j qualifies v) and
    (v supports this) $
END
```

### 3.5. Replay of Argumentation

Retracing various steps in the argumentation process will be very valuable. This may be used to take corrective action or to train new participants in a project or to facilitate understanding of critical actions. Also, as the various choices considered are still available for review, such a replay could also help trigger alternate actions.

The replay can be done in a chronological fashion. In other words, an argumentation can be played out as it happened during the original session. This will be useful as a training mechanism for newcomers, or a process toward creating organizational knowledge.

As ARBAS views negotiations as a sequence of actions

in a temporal sequence, such a facility will be very critical to support discussions. In our system, for each assertion in the knowledge base, a timestamp is created. This can be used to trace the evolution of actions.

The replay can also be done in a dependency directed way. For example, the users may be interested in tracing the repercussions of a critical justification on various components of an action. The user may want to identify why a particular action (say, DESIGN2 in our example) was not pursued. Pomerol et al [8] argue that such "reasoning backwards from outputs" is more than the traditional "what-if analysis involving reasoning forward from data. In such a replay, we start with the goals (here, ACTIONS) and ascertain whether it can be carried out by changing the decision inputs (such as the belief in justifications).

#### 4. Related Work

Several models and systems that support the acquisition and use of argumentation have been developed recently. The gIBIS system [8] and others that are inspired by it [e.g., IBE [16] represent the most popular approach of capturing conversations among the stakeholders involved in complex problem solving activities. These systems provide a hypertext interface to the IBIS model. A limitation of the IBIS model on which these systems are based is the lack of representation of inputs and outcomes of argumentation. ARBAS specifically addresses this issue by relating argumentation views to the action triplet that is the focus of the discussion. Further, resources that are the inputs and outputs of the actions are also explicitly represented. This approach is complementary to efforts such as in the REMAP project [9], which extend the IBIS model to relate argumentation to its context. Unlike REMAP which provides generic links to contextual objects, we focus on action-resource centered argumentation.

Many groupware systems (such as GroupSystems, VisionQuest) focus primarily on capturing informal knowledge exchanged during group discussions. Systems such as Constellations [10] facilitate access to informal knowledge by chunking multimedia information. However, the need to integrate informal information in hypertext with formal models has also been suggested by Bhargava et al [2]. A primary reason for providing a formal representation of the argumentation process (whenever feasible) is to facilitate automated reasoning with the captured knowledge. Our work, therefore, takes a comprehensive approach to dealing with informal information advocating the creation of formal models to facilitate better integration. Further, instead of maintaining passive argumentation knowledge (as in IBIS

and QOC), our focus is on providing intelligent support by maintaining an active history of actions and their contexts. This approach is similar in spirit to that of the SIBYL tool that provides various services to use decision rationale [6]. Whereas SYBYL is based on DRL, a generic language for decision rationale, our focus is on action-resource based argumentation. We also suggest a tight integration of formal and informal components of argumentation knowledge.

#### 5. Future Work

A field study on capturing and using design rationale [14] demonstrates the feasibility and usefulness of capturing semi-structured knowledge about design decision making in large system development projects. However, the capture and maintenance of argumentation knowledge can be very expensive. Therefore, non-intrusive and easy capture of this information is essential. We are currently investigating the capture of decision rationale from electronic mail exchanges. Future work will include the machine learning techniques to classify outputs from group support to at least partially automate the generation of this knowledge. We are also developing a set of ad-hoc queries that could support different stakeholder needs for access to relevant parts of the argumentation knowledge. The knowledge base of *semi-structured* experiential knowledge can provide lower level learning [13] by helping the users detect standards and deviations from them in solving problems. This knowledge also lends itself well to the use of induction and case based reasoning approaches to learning. The empirical evaluation of the usefulness of our approach is a topic of future research. Our evaluation would focus on the feasibility of capturing and maintaining argumentation knowledge in system development activities.

#### References

- [1] Binbasioglu, M, Bui, T., and Ma, P-c, "An action resource language for argumentation: The case of softwood lumber negotiation", In Proceedings of the 28<sup>th</sup> Hawaii international conference on information systems, January 1995.
- [2] H. Bhargava, R. Krishnan, and A. Whinston, "On integrating collaboration and decision technologies," *Journal of Organizational Computing*, vol. 3 (4), 1994, pp. 297-317.
- [3] J. Conklin and M. Begeman, "gIBIS: A Hypertext Tool for Exploratory Policy Discussion", *ACM transactions on Office Information Systems*, Vol. 6, 1988, pp. 303-331.
- [4] M. Jarke, "ConceptBase v 3.0 manual", Tech. Rep. MIP-9106, University of Passau, Germany, 1993.
- [5] M. Lease, M. Lively and J. Leggett, "Using an issue-based hypertext system to capture the software life-cycle process", *Hypermedia*, Vol. 2, No. 1, 1990

- [6] J. Lee, "SIBYL: A Qualitative Decision Management System", In *Artificial Intelligence at MIT: Expanding Frontiers*, Edited by P. Winston and S. Shellard, MIT Press, Cambridge, MA, 1990.
- [7] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis, "Telos: Representing knowledge about information systems," *ACM transactions on information systems*, vol. 8, October 1990, pp. 325-362.
- [8] J.C. Pomerol, B. Roy, and C. Rosenthal-Sabroux, "Developing an "intelligent" DSS for the multicriteria evaluation of Railway Timetables: Problems and Issues," in *Proceedings of the Third international Conference of the International Society for Decision Support Systems*, vol. 1. Hong Kong, 1995, pp. 161-172.
- [9] B. Ramesh and V. Dhar, "Supporting Systems Development by Capturing Deliberations during Requirements Engineering," *IEEE Transactions On Software Engineering*, vol. 18, 1992, pp. 498-510.
- [10] V. S. Rao and R. Goldman-Segall, Capturing Stories in Organizational Memory Systems: The role of Multimedia, In *Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, vol. III. 1995, pp. 333-341.
- [11] B. Silverman, "Unifying Expert Systems and the Decision Sciences," *Operations Research*, vol. 42 (3), pp. 393-413, 1994.
- [12] Stefik, M. et al. "WYSIWIS Revisited: Early experiences with multi-user interfaces", *ACM Transactions on Office Information Systems*, Vol. 5, April 1987, pp. 147-167.
- [13] EW Stein, and V. Zwass, Actualizing Organizational Memory with Information Systems, *Information Systems Research*, vol. 6, no. 2, 1995, pp. 85-116.
- [14] K.B. Yakemovic and J. Conklin, Report on a Development Project Use of Issue-Based Information Systems, In *proceedings of the Conference on Computer Supported Cooperative Work*, 1990, pp. 105-118.

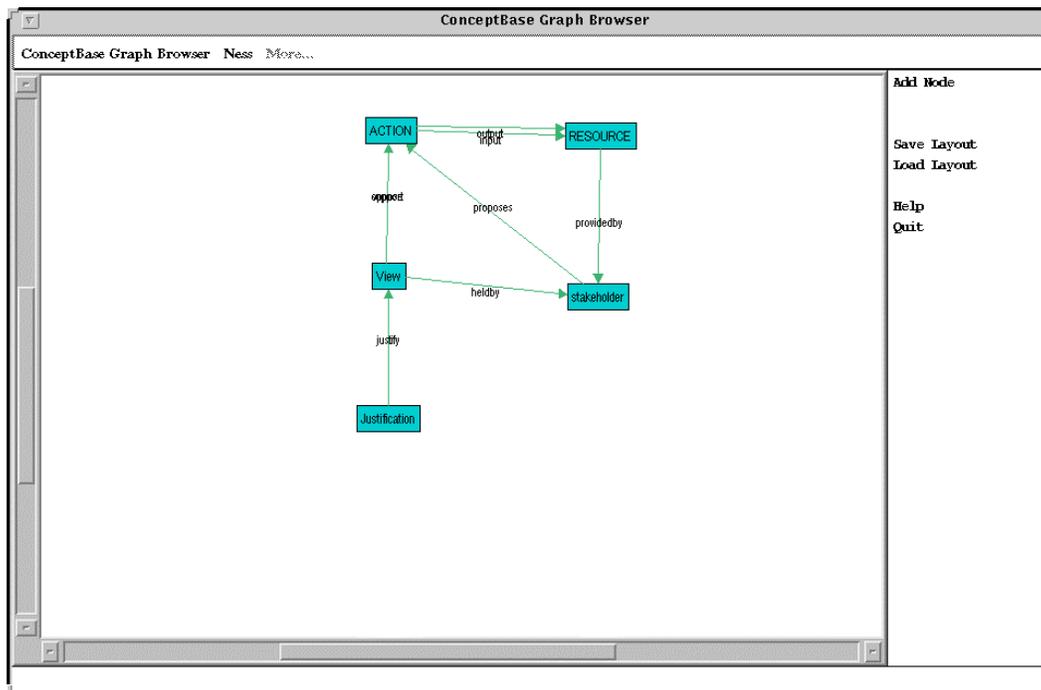


Figure 1: ARBAS implementation in ConceptBase

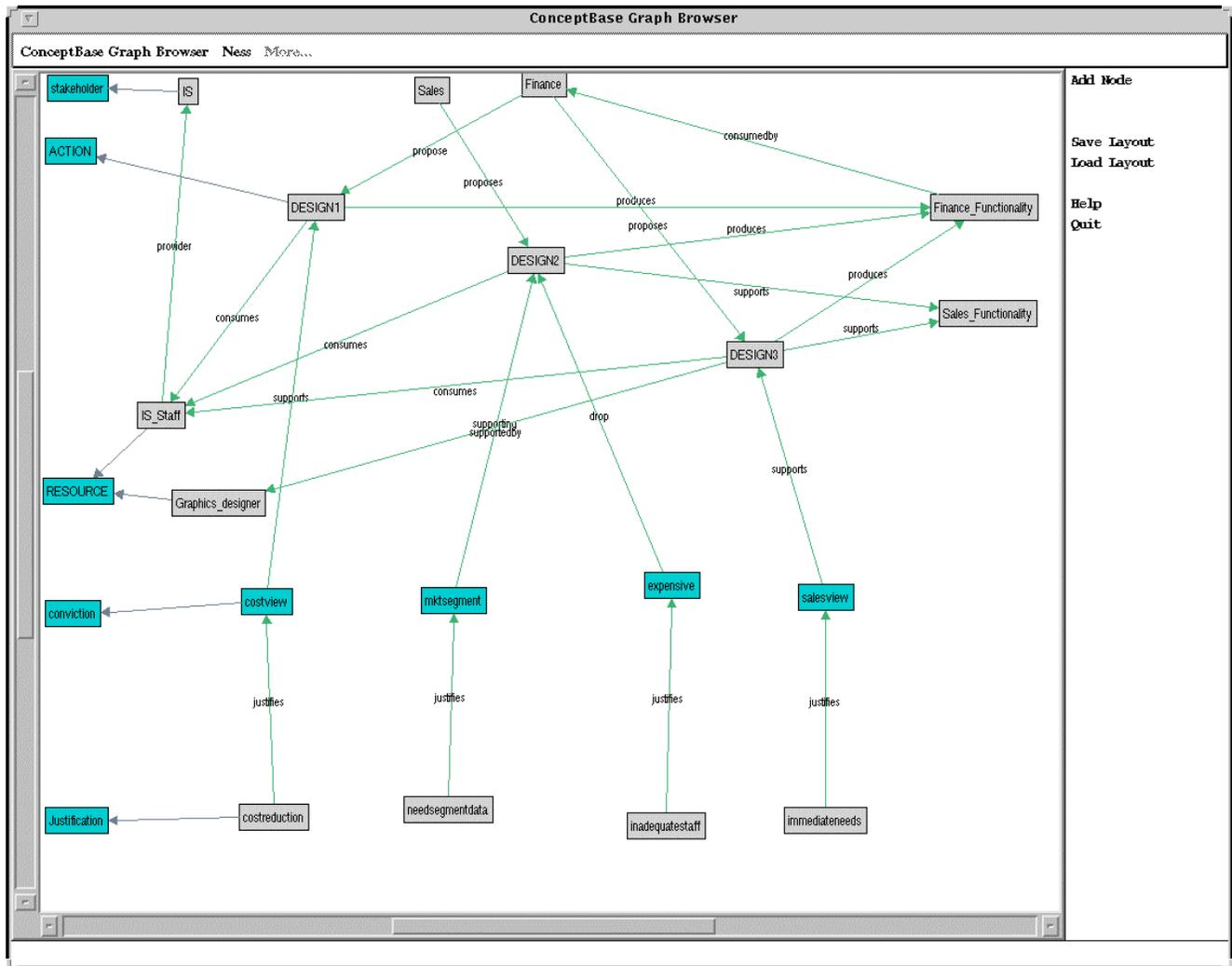


Figure 2: An ARBAS Session