Cosmos: A System for Supporting Engineering Negotiation

William Mark
Jon Dukes-Schlossberg

Lockheed Artificial Intelligence Center
3251 Hanover St. O/96-20 B/254F
Palo Alto, Ca. 94304
(415) 354-5260
mark@sumex.stanford.edu

Abstract

Large scale engineering projects are created by teams of cooperating engineers who must share knowledge about the project as it evolves. Sharing of engineering knowledge is actually made *more* difficult by modern engineering environments. First, the computerized engineering environment requires that much of the knowledge sharing be on a tool-to-tool basis, rather than human-to-human. Computer tools (3-D modelers, analysis and simulation tools, etc.) have become the locus of much of the engineering information for a project. These tools embody engineering assumptions and methods that are not understood in detail by their user engineers. Moreover, the tools use specialized internal representations that are not understood by other tools. Second, the connectivity enabled by the modern networked engineering environment greatly increase the complexity of the interaction environment. It is virtually impossible for engineers to know who is likely to be impacted by their decisions, and what issues they need to negotiate. The Cosmos project is part of a collection of research efforts that is creating technology to allow engineers to share knowledge about a design through their tools: when engineers modify a design, the tools they are using automatically provide relevant updates to other engineers whose work is affected by the change. Cosmos's role is to support engineering negotiation, illustrated here in the domain of active control of spaceborne structures. The paper describes the Cosmos Phase I implementation, which provides negotiation support for engineers from different disciplines cooperating on a design; lessons learned from Phase I, and our current activities in building the Phase II implementation.

1. Introduction

Large scale engineering is distributed in terms of both space (teams are often geographically disparate) and time (decision making is often spread out over long periods of time -- sometimes years). But engineers must share information in order to contribute to a project. Engineers do not make decisions in a vacuum. Each decision is made in the context of a shared specification and implementation; and each decision must usually be negotiated with other engineers. Traditionally, the information that engineers needed to share was in their heads and in documents. Cooperation among engineers was achieved by face-to-face meetings and exchange of documents.

Cosmos is the effort of a team that includes Lori Ogata, Ray Johnson, James Hyde, Bob MacGregor, and Steve Mudry.

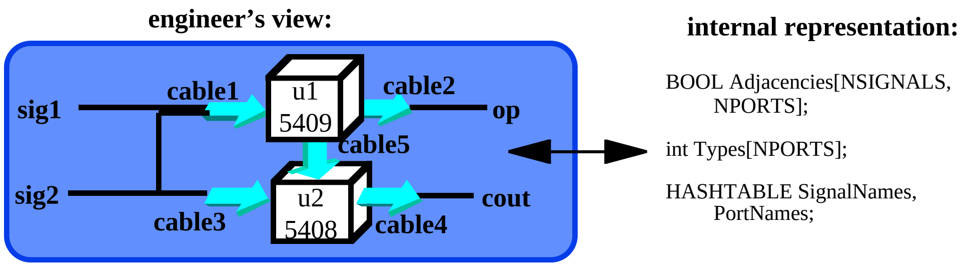
Now virtually all advanced engineering takes place using computer tools that support modeling, analysis, simulation, process planning, etc. Much of the information that the engineers develop about a project is contained in their computer tools. Engineering collaboration must now involve the sharing of computer-based information. The past 20 years has seen rapid advances in computer aided engineering technology, causing the productivity of individual engineers to rise dramatically. Unfortunately, the computerization of engineering has actually made *sharing* of engineering information more difficult in at least two ways:

- Much of the information that must be shared resides only in the specialized representations of different computer tools, each of which solves only part of an engineering problem. Engineers cannot be aware of the information requirements and capabilities of the tools, and therefore cannot be responsible for managing tool-to-tool information sharing.

- The large, distributed collaborative environment enabled by modern computation technology is complex and dynamic. Normal engineering processes such as discussion and negotiation require computational support.

1.1 Direct Tool-to-Tool Sharing

Engineering collaboration requires sharing of analyses, the assumptions underlying these analyses, and the specific methods used to compute the analyses. Much of this information is not designed for presentation to the engineers. Say that a cable design tool is being used to automatically route a cable within an electromechanical device. In the left part of Figure 1 we see the engineer's view of the information: a graphic that is meaningful to engineers in terms of their external world. In the right part of the figure we see the tool's representation of the information: a collection of data structures and algorithms that have been specialized to calculate routings and to produce the desired graphics. The tool can show the result of its routing as a graphic display, and engineers will be able to discuss this display in a video conference -- but this is not the same as sharing the analysis that underlies the display. The assumptions on which the routing is based are contained in the details of the software. They are certainly not apparent from the graphic output, and the engineers are probably not aware of all of them.



```
BOOL Adjacencies[NSIGNALS,
      NPORTS];

int Types[NPORTS];

HASHTABLE SignalNames,
      PortNames;
```

Figure 1: Tools keep much of their knowledge to themselves

The engineer sees a cable routing generated from data structures in the tool; the underlying assumptions and methods used to determine the routing stay buried in the software.

Furthermore, the routing tool solves only part of the problem. Its results are most useful to other tools, which can incorporate those partial results into their analyses and simulations. It is the combination of the results of several tools that is of interest to engineers. For example, suppose that an engineer proposes moving the cable. One of the key reasons to share engineering information is to identify the impact of proposed engineering changes. However, the data residing in any individual tool does not represent enough information to determine the impact of the proposed change. The computer tool used to make the proposed change probably has a geometric representation of the cable, i.e., its data represents the cable as a collection of points in space. But determining the impact of moving the cable requires interaction with the structural analysis tool: moving the cable may have structural impact on other parts of the device. Moving the cable may also have thermal, electromagnetic, and other impact on other parts of the device. Other tools will have appropriate thermal, electromagnetic, and other representations of the cable. It is the sum of the results of all of the relevant tools that contains the information that engineers need to collaborate.

The engineers cannot be responsible for incorporating the results of one tool into the other ones that need it. Since the underlying assumptions and methods of the tools are not easily accessible to the human engineers, they cannot know the consequences of feeding the results of one tool into another. The tools must exchange information with each other, and must incorporate this exchanged information into their own analyses or simulations without human intervention.

1.2 Automated Information Flow and Negotiation Management

The collaborative environment is a complex place, with participants and roles continually evolving. It is difficult to determine which engineers should receive which information, in what form, and with what frequency. This includes the problem of deciding which tools need to exchange information in order to support a specific collaboration. For both human-to-human and tool-to-tool sharing, it is very difficult for individual participants to know who else in the environment should be receiving their ideas and results, and who else's ideas and results they should be receiving. If the cable design example mentioned above were part of a large, distributed project, it would simply not be feasible for the engineer who wanted to move the cable to know which engineers are likely to be affected by the change, and what information their tools need to determine the effects of the analysis. Similarly, on a large project it is simply not feasible for engineers to go through all of the proposed changes to see which ones are likely to affect them.

It is therefore essential that the environment provide some help in automatically determining interests and routing information to interested parties. This must be more than a simple exchange of messages. Engineers do not simply exchange information; they negotiate, explore alternatives, look at previous designs, and so on. The environment must take a role in ensuring that all of the engineers required for a particular decision are involved in the decision process, and that they (and their tools) have the right information at the right time.

In summary, then, in the collaborative engineering environments that are now evolving, processes like negotiation require computational support, first, because much of the information that is being negotiated is embedded in the tools, and second, because the number and diversity of those tools creates an information environment that is too complex for effective human comprehension and management.

2. Tool Integration at the Knowledge Level

Supporting engineering negotiation in a distributed computational environment requires a new tool integration technology: The specialized representations of individual tools must be linked to a shared, explicitly represented terminology; tools must interact via an infrastructure that supports incremental exchange of knowledge expressed in this terminology; and this incremental exchange process must be managed to allow distributed engineers to participate in the same engineering process.

Conventional approaches to integrating engineering tools depend on standardized data structures and a unified engineering model, both of which require substantial (and often unrealistic) *a priori* commitments from tool developers. A collection of research efforts [PACT, SHADE, SHARE] is creating computational infrastructure technology to allow engineers to share information at the *knowledge* level: the environment provides a shared knowledge base and mechanisms for managing information flow in terms of that knowledge base. Engineers working on cable design can communicate with each other because they share a common basis of knowledge about cables. They know that cables contain wires or fibers, and thus carry signals; that they heat up, have weight and volume, cannot be stretched too far or bent too sharply, attach to ports, and so on. This knowledge is much more than a dictionary; it is a knowledge structure or *ontology* [Gruber] that defines the key objects and actions in the domain, and the key relationships among them.

Knowledge level integration is achieved not at the level of data structures and models *within* the tools, but at the level of a shared communication language (that is, a system of grammar, vocabulary, and meanings) *among* tools. The vocabulary and meanings of the language are defined by the ontology, which needs to represent only information that is useful to other tools -- a small subset of all the data structures within the tool. Standards for knowledge level inter-tool languages are starting to emerge, e.g., KIF (Knowledge Interchange Format) [KIF] and KQML (Knowledge Query and Manipulation Language) [KQML].

The environment must also provide support for managing the communication process. As we have seen, it is not feasible to make individual engineers responsible for understanding who needs to know what -- they cannot possibly keep track of the range of interests and capabilities in a dynamically changing environment, especially for a project that lasts over months or years. Instead, the environment must provide automated information routing.

If information routing is to be automatic, it must be based on the *content* of the information. The knowledge level inter-tool language makes this possible. All communication in the environment is in terms of KIF/KQML message interchange. Engineers and tool developers create messages to inform the environment of their interests and the capabilities of their tools (expressed in KIF). The meanings of the KIF terms are defined in the ontology. These messages also specify (in KQML) the way in which information is to be given or received (via broadcast or direct connection, intermittently or periodically, etc.). If the situation changes (e.g., interests change or participants change their status in the environment), update messages are sent. Because these messages are expressed in the formal declarative KIF/KQML language, the meaning does not depend on the different representations of different tools, and can therefore be shared among all tools. Moreover, because the meaning is expressed in this declarative manner, special automatic routing agents in the environment can examine the content of the messages and route them selectively to participants that have expressed interest in that kind of information.

But even automatic routing is not the whole solution. When engineers make decisions that affect the work of other engineers (which means most of their decisions), they must participate in a *negotiation* that allows other engineers to assess the impact of the proposed decision. The engineers whose work is affected usually offer alternatives and discuss trade-offs. Engineers express new decisions and offer alternatives in terms of changes to component descriptions. A key part of the engineering negotiation process is to determine which other components depend on the component being changed, who is responsible for these components, which of them might have to change, and how difficult it will be for them to change.

When projects reach large scale (thousands of interacting components), these questions are almost impossible to answer without computational aids. In the absence of automated technology, engineering negotiation must rely on informal understandings (which works very well for small efforts, but not for large ones) or on rigid controls (which reduces engineering productivity). A few research efforts are beginning to create automated information flow support technology by incorporating "coordination models" into the environment (e.g., [Petrie]). That is, interactions among participants are guided by a built-in model that provides *a priori* structuring of the interaction process based on theories of how engineering interaction works or should work. While this approach provides much needed organization for the information flow in a collaborative environment, the resulting organization does not embody enough knowledge to support activities like negotiation.

Cosmos is focusing on supporting the negotiation process by which distributed engineers agree on engineering decisions. Engineering negotiation is a sophisticated process, requiring knowledge about trade-off strategy as well as in-depth engineering knowledge. Cosmos does not perform the negotiation itself. Instead, Cosmos receives proposed engineering changes, automatically determines ramifications of the proposed changes, and then presents a visualization of these ramifications to all engineers whose work will be affected by the proposed changes. This visualization forms a shared context among these engineers for negotiating about the proposed change.

3. Cosmos Overview

The Cosmos work was motivated by our experience in implementing a demonstration of the knowledge level integration technology as part of PACT (Palo Alto Collaborative Testbed) [PACT]. PACT experiments feature the distributed engineering and simulation of the Planar Manipulator, a small robotic device for positioning an object on a planar surface. Responsibility for engineering and simulation of the device has been parceled out among four independently developed systems: Designworld handles the electronic circuitry of the Planar Manipulator, NVisage the software controller, Next-Cut the physical mechanism, and DME the power train.

Simulation and engineering in this environment requires the interoperation of these systems. For example, during a PACT simulation, DME discovers that the motor would burn out if it were subjected to the simulated loads. It alerts the engineers to that fact, causing them to replace the existing motor with a larger one in the design. The engineer chooses a larger motor, and Next-Cut notices that this motor requires a larger drive shaft, and that a larger shaft hole must be engineered in the motor housing. Next-Cut then goes on to create a process plan to machine that hole.

This interaction requires a great deal of knowledge sharing. But so far in PACT this knowledge sharing has occurred "off line". What went on behind the scenes to develop this scenario, and is certainly not represented in computational form in PACT at all, was a careful negotiation to ascertain the feasibility and advisability of this set of engineering decisions. All of the human engineers got together to discuss the ramifications of the larger motor, determining for example that the motor housing would not have to be enlarged and that the added weight of the new motor would not affect operation of the manipulator's effector arms. These findings were critical: enlarging the motor housing or changing the dynamics of the arms would have required major changes in the rest of the manipulator design. The proposed change was allowed, but only because its effect on the existing design was determined to be acceptable. It is this process that we are bringing into the computational framework via Cosmos.

In particular, Cosmos provides automated reasoning to support engineering negotiation. This reasoning involves management of *commitment* constraints, a subset of engineering constraints that determine whether a particular component fits into a particular design. The commitment types for any particular domain represent the constraints that are known to be relevant in determining the implications of component descriptions on each other. For example, in a mechanical domain, physical linkages, spatial relationships, and functional roles define commitments. In an electrical domain, commitments are concerned with connectivity, physical configuration, thermal, and radiation characteristics. In software, input/output, data access requirements, and control relationships represent commitments (for detailed examples, see [Mark]).

Commitments define a concept of locality with respect to engineering change: when a component description changes, the other components specified in its commitments are the ones affected. Commitment relationships thus isolate a subset of the design that needs to be considered for a single negotiation. In Cosmos, to ensure that components meet commitments, component descriptions are organized dynamically into a coherent representational framework that represents the "current design" as it evolves. This coherent framework is the *engineering model.* Cosmos represents the engineering model in Loom [MacGregor], and uses Loom to perform subsumption and constraint propagation reasoning over the engineering model to ensure that components meet each others' commitments in the face of change.

Whenever a new engineering decision is proposed via one of the tools in the environment, Cosmos (1) determines the *scope of impact* of a decision in order to provide appropriate context for the engineers involved in negotiating a change; (2) produces a *visualization* of the scope of impact; and (3) presents this visualization to all of the *stake holders* to give them an explicit shared context for negotiation.

Cosmos assesses the impact of a proposed change via qualitative simulation [Forbus] on the commitment relationships that include the changed element. For example, engineers may want to determine whether a change in gimbal mass will cause the stress on a strut to exceed the maximum acceptable stress. In general, Cosmos will not have enough detail to calculate the stress directly. It will, however, know that the stress varies proportionately with the mass and can thus determine whether the maximum acceptable stress level would be affected by the change in mass, i.e., whether the stress level is within the scope of impact of the proposed change in mass.

This kind of reasoning results in the determination of the neighborhood of interrelated engineering elements that are relevant to this particular decision, which defines the scope of impact. Cosmos produces a visualization of this scope of impact neighborhood in the form of a "network browser" showing the relevant components and their interrelationships (see Figure 6 below).

Cosmos shows this visualization to the engineers who proposed the change in order to provide immediate feedback: for example, if the engineers specify a change that leads to onerous commitments, they may wish to reconsider their proposal---one of the most powerful reasons not to change a design (or to change it) is to make it fit in with previous decisions.

So far the scope of impact visualization has provided feedback to the original engineers based on the current engineering model---but this model will change as a result of the negotiation required to incorporate the proposed change. To help manage this process of change in a distributed setting, Cosmos uses the scope of impact visualization to provide a context for negotiation with other engineers. Therefore, once the original engineers are convinced that they actually want to propose the change, Cosmos presents the scope of impact visualization to the other engineers whose components are within the scope of impact---the "stake holders." Rather than reacting to the proposed change in isolation, all of the parties involved share the same scope of impact visualization context for exploring decisions. As different engineering groups propose reactions to the original change, Cosmos distributes the proposed updates to the shared context to all of the stake holders. The scope of impact visualization thus serves as a shared dynamic context in which all stake holders make negotiation decisions.

Figure 2 depicts the architecture we have used for our Phase I implementation. Two engineers, a gimbal engineer and a layout engineer, are working on the design of a spacecraft structure via their (very different) computer tools. The tools are linked together through the knowledge sharing infrastructure described in Section 2. Tools are attached to the infrastructure via "wrappers", which include translators between the tool's internal language and the KIF/KQML language of the infrastructure. Tools exchange information through their wrappers in terms of KIF/KQML messages. These messages are automatically routed to the tools of other engineers that have declared interest in the kind of information found in the content part of the message (see [SHADE]).

Cosmos is also part of the infrastructure, acting as a mediator [Wiederhold] for determining information that tools *should* be interested in, beyond the interests they have declared[1]. Cosmos examines all messages that suggest engineering changes (these are automatically routed to Cosmos because Cosmos has declared interest in them). Cosmos uses its engineering model and commitment-based reasoning to determine the scope of the impact of the change. This scope of impact is then used to produce a visualization. Finally, this information is packaged into a KIF/KQML message, which is automatically routed to all of the tools that have declared an interest in any of the elements of the scope of impact -- a more complete set than those which had expressed interest in the original change message itself. This forms the set of stake holders who must be involved in the negotiation. The stake holders then negotiate on the basis of a shared context -- the visualized scope of impact that they now share. Cosmos monitors the ongoing negotiation, updating the shared visualization as decisions are proposed and made. All information is exchanged using the message passing protocol.

One of the goals of Cosmos, along with the other elements of the infrastructure, is to allow the inclusion of tools into the infrastructure with a minimum amount of work required on the part of tool developers or user engineers. The only additional burden added by Cosmos is a simple I/O manager as part of the wrapper, used to manage interaction with the on-screen visualization.

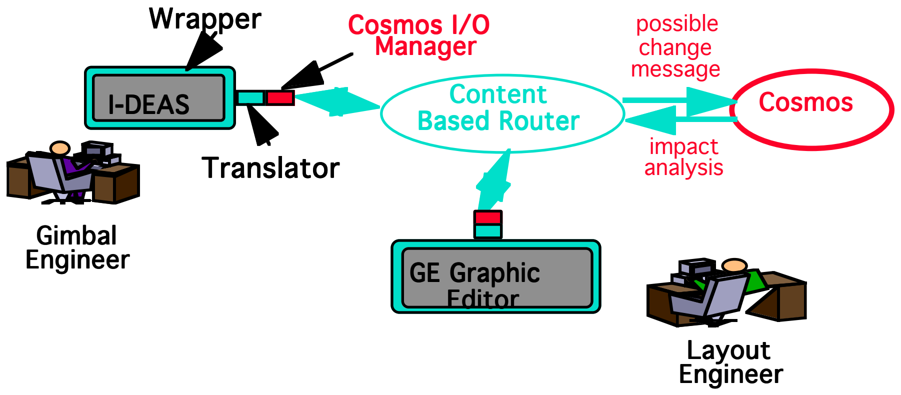


Figure 2. Cosmos Phase I Implementation Architecture

[1]A travel agent is a familiar example of a mediator, using his or her knowledge of travel "capabilities" (routings, fares, packages, etc.) and client needs to provide clients with information that they were unaware of, but should have in order to make their decisions.

4. MACE

The first real test domain for Cosmos has been an experimental spaceflight program, the Mid-deck Active Control Experiment (MACE), scheduled to be flown on the Space Shuttle mid-deck in 1994. MACE is a joint MIT / NASA / Lockheed research effort to investigate problems common to flexible spacecraft in order to aid in the engineering of future satellites. Specifically, MACE will use active control to allow multiple payloads (e.g., multiple telescopes) to coexist on a single satellite bus structure. Without active control, when one of the telescopes receives a ground instruction to point to a new location, ensuing vibrations in the bus structure caused by the moving telescope would significantly disrupt the other telescopes.

The MACE engineering model maintained by Cosmos represents the components and interrelationships defined in the current MACE design, including the current values for engineering quantities (e.g., masses, forces, stresses, etc.) within the design. All MACE components and relationships are defined by one of the ontologies used by Cosmos. These ontologies contain all of the engineering terms and their interrelationships known to Cosmos.

Figure 3 shows a fragment of the MACE engineering model, and its relationship with (an even smaller fragment of) the ontology of structures. Strut1 is an element of the MACE design that is a kind of cylindrical object. By virtue of the definition of cylindrical object in the structures ontology (fragment shown in bold), the MACE engineering model inherits a set of interrelationships that define key engineering quantities like "strut stress". Depending on the detail in which interrelationships have been modeled in the ontologies available to Cosmos, engineering model links vary from actual equations to qualitative relationships such as "directly linearly proportional", "directly non-linearly proportional", or simply "related". Note that the concepts in the ontologies are generic, and only have to be modeled once to cover all designs within a specific engineering discipline.

The concepts in the MACE engineering model are specific to that design. For example, the MACE bus consists of a specific set of struts and gimbals; the stress on strut in the MACE design is governed by an "acceptability constraint", which is in turn determined by a safety factor, and so on. Each of the terms in the model like strut, gimbal, and safety factor is defined in a generic ontology, however the specific configuration of these terms, and specific values of engineering quantities are specific to the MACE engineering model. In fact, any real design will be characterized by a collection of engineering models over time, since changes in the design lead to changes in the engineering model. Cosmos determines scope of impact based on the existing engineering model, and changes the model to reflect agreed-upon engineering changes, in turn creating the next "existing" model.

Commitments are defined within the ontologies (since they are properties of the engineering discipline), and are thus inherited by the engineering models. All the interrelationships shown in Figure 3 are commitments.

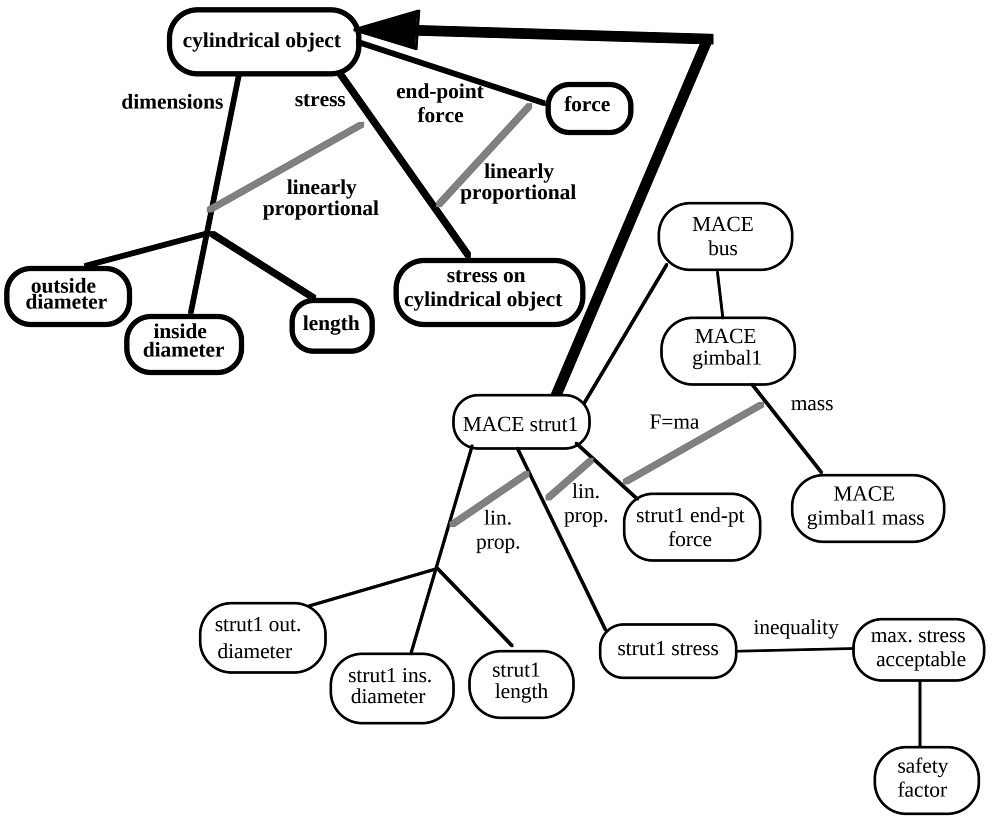


Figure 3. Fragments of the "structures" ontology and MACE engineering model

Relationships in the MACE engineering model are inherited from the ontology of structures (shown in bold above). Since "MACE strut1" is a cylindrical object (the dark arrow indicates a subsumption relationship), all other relationships are inherited (solid lines are "roles" or "slots"; dashed lines are relationships between roles).

5. Example of Cosmos Interaction

In a scenario of Cosmos support, a gimbal engineer learns that the slew speed of the MACE gimbals must be increased (a payload is mounted on the MACE bus structure by a gimbal, which is responsible for pointing the payload via "slew" and other commanded motions). To accomplish the faster slew speed, the gimbal engineer considers using a different, larger inside rotor within the gimbal . The engineer uses his tool (the IDEAS solid modeling and structural analysis system) to select a new inside rotor and see if it will do the job. In the local world of the gimbal engineer, the problem is now solved. However, because the gimbal is only one part of the larger MACE design, the gimbal engineer must consider the impact of his decision on other engineers: The engineer must choose a replacement rotor that minimizes impact on the rest of the design, in order to facilitate acceptance of this engineering change. That is, finding a rotor replacement that fits in with the rest of the design is just as much part of the engineer's problem as achieving greater slewing speed.

It is therefore in the engineer's interest to explore the impact of his decision (and perhaps try alternatives) before actually proposing a change to the rest of the design team. The engineer can use Cosmos to help him with this initial assessment of the impact of his proposed change. The engineer is constantly interacting with IDEAS, making "edit-level" changes, running analyses, etc. Cosmos only enters the picture when the engineer reaches the stage of having made a provisional engineering decision, which he signals by essentially saying "done" to the tool (the exact interaction is dependent on the tool, and is monitored by the Cosmos I/O manager). Cosmos then provides the impact analysis automatically, as part of the infrastructure. The engineer does not have to "call" Cosmos, or learn any new commands.

When the engineer signals "done", the Cosmos I/O manager tells the wrapper associated with the IDEAS tool to format a *possible change* message that contains the engineering parameters (gimbal mass, geometry, and dynamics) that have changed since the last "done" signal. This *possible change* message is then broadcast over the infrastructure and forwarded to Cosmos by the content-based router, since Cosmos has posted an interest in all engineering change messages (see Figure 4). Cosmos then examines the current MACE engineering model to determine the scope of impact of this change.

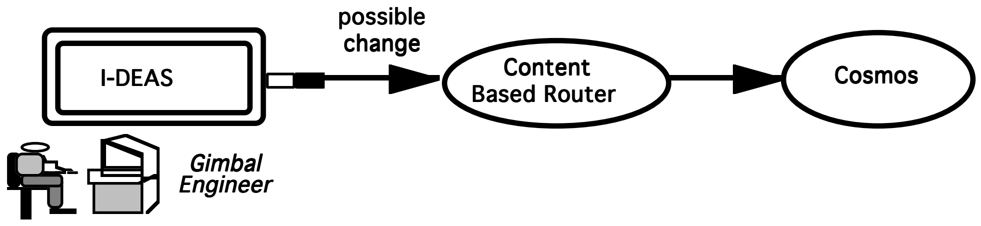


Figure 4. Step 1: Gimbal engineer explores a possible engineering change

The gimbal engineer uses I-DEAS to define a possible engineering change. Cosmos has posted an interest with the content-based router so that it will receive notification of such changes and be able to provide impact analysis.

When Cosmos receives a possible change message, it creates a new context in which it can reason about this change without affecting the agreed-upon engineering model. Within this new context, the engineering model is updated with the new values for the possible change. In this case, the new larger rotor has caused a 10% increase in the gimbal mass value. Cosmos explores the commitments relating "gimbal mass" to the rest of the engineering model in order to identify the scope of impact within the model. Some of these commitments are shown in the model fragment of Figure 3. For example, following one of the commitments from gimbal mass, Cosmos uses qualitative sensitivity analysis to determine that, given the F = ma proportionality relationship between gimbal mass and strut end-point force, the strut end-point force quantity must increase by the same percentage the mass has increased i.e., 10%.

Cosmos goes on to explore the commitment between strut end-point force and strut stress. It determines that this value must also increase 10%, since strut stress is directly linearly proportional to strut end-point force. Next it finds that strut stress is constrained to be less than the maximum acceptable stress. If the engineering model contains the actual values of these quantities (along with their interrelationships, as shown here), Cosmos can determine whether the "acceptability" constraint would be violated by this change. Otherwise, Cosmos flags it as a potentially violated constraint (e.g., here the potential violation would be that if the maximum acceptable stress is less than 10% of the previous stress value, the acceptability condition would be violated). This commitment evaluation process theoretically continues until it "grounds out" in leaf nodes in the model. But Cosmos produces the scope of impact for visualization a "page at a time", allowing commitment evaluation to be a lazy background process.

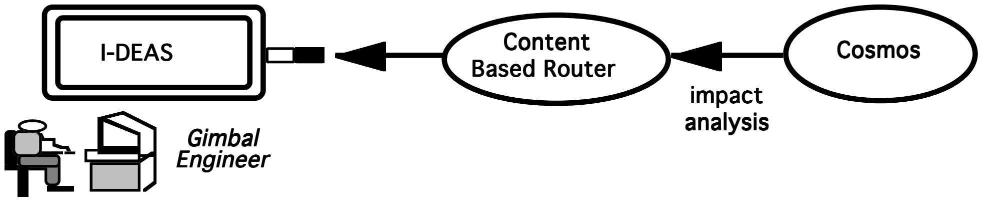


Figure 5. Step 2: Cosmos provides scope of impact

Cosmos provides the scope of impact of the possible change to the gimbal engineer, who decides to try a different alternative.

"Pages" are defined by proximity to violated or potentially violated constraints[2]. When Cosmos computes a page's worth of commitment evaluation information, it packages the information as an impact analysis message for routing back to the gimbal engineer (see Figure 5). The Cosmos I/O manager for the gimbal engineer's IDEAS tool then produces the visualization for the engineer (see Figure 6). This visualization gives the engineer immediate visual feedback on the ramifications of the change: each commitment must be met, which means that the designated other parts of the design must be made compatible with the change in gimbal mass.

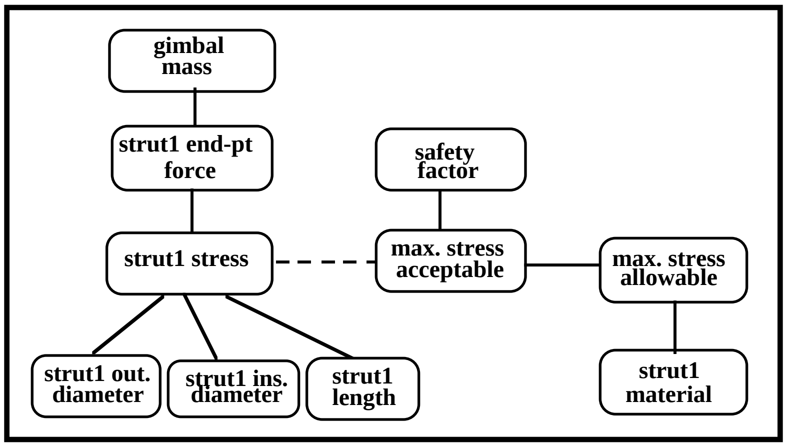


Figure 6. Visualization provided by Cosmos

Links represent constraints; the dashed link (colored red on the actual Cosmos display) represents the constraint violation or potential violation.

[2]The definition of proximity here is somewhat complicated, as it depends on the characteristics of the engineering model in Loom (number of roles per concept, etc.), the fact that not all Loom concepts and roles are visualized for the user (only those with specific relationships to pre-marked "visualizable" ontology elements are visualized), and user interface ideas of what size diagram is effectively visualizable. In any case, paging here is a computational convenience, not a theoretical statement.

The visualization shows the engineer the context for negotiation with other engineers: a constraint has been violated, and one or more of the design elements shown in the visualization must be changed. Some of these elements are under the control of the gimbal engineer (e.g., the gimbal mass), while others are not (e.g., the strut diameters). The gimbal engineer can thus see the context in which the negotiation will take place, and make a judgement as to whether the change is worth pursuing, or whether an alternative should be tried. This page represents only one aspect of the context; the engineer may have to look at several pages to get the full picture (for simplicity, only this page will be discussed here).

In this case, the gimbal engineer decides from the feedback that other engineers will have a hard time accommodating the change in gimbal mass. He therefore uses IDEAS to explore an alternative: choosing a different inside rotor, one that is not optimal for his design, but is less massive. When he signals "done", the Cosmos I/O manager once again tells the IDEAS wrapper to send a possible change message, which is received by Cosmos. Cosmos creates a new context, evaluates commitments, and sends back a new visualization page (see Figure 7).

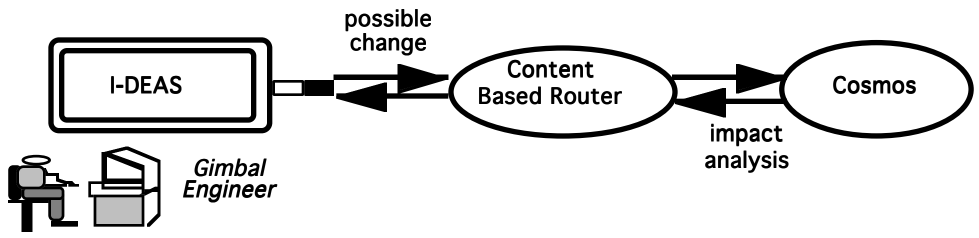


Figure 7. Step 3: Engineer tries again

Gimbal engineer tries an alternative possible change and receives a new visualization.

This process of trying possible changes and receiving visualization feedback may go through several iterations before the gimbal engineer believes that a given alternative meets the required functionality and will not be too onerous (or seems to be the least onerous) for other engineers to deal with. In this case, this occurs on the second iteration. The gimbal engineer has been using the negotiation context provided by Cosmos to determine how other engineers might react to his suggested change. Now he must actually negotiate that with the other engineers affected by this change, the stake holders.

The advantage of the Cosmos approach is that along with the possible change, the gimbal engineer can share with the stake holders the context he has been using to explore the ramifications of that change. When the gimbal engineer decides to actually bring the other stake holders into the negotiation, he signals through the IDEAS tool that he is ready to actually propose the change. The Cosmos I/O manager then tells the IDEAS wrapper to create a *proposed change* message, which is again received by Cosmos (see Figure 8). This time Cosmos will send the proposed change and scope of impact pages to all of the stake holders, not just the gimbal engineer. Cosmos and the content-based router determine the stake holders automatically: the scope of impact shows which MACE elements are affected by the change, and the content-based router knows the interests of the various MACE engineers (based on previously collected subscription information, see [SHADE]). Cosmos therefore send the proposed change, in context, to all of the stake holders. Together this information is used to make sure that all of the affected engineers receive the proposed change and the relevant scope of impact pages.

The scope of impact can now be used by the other engineers as a context for agreeing with the change (if it has only limited effects on their part of the design), or disagreeing with it (e.g., if it would require them to change a critical component). They can also agree with the change contingent on a further change being made.

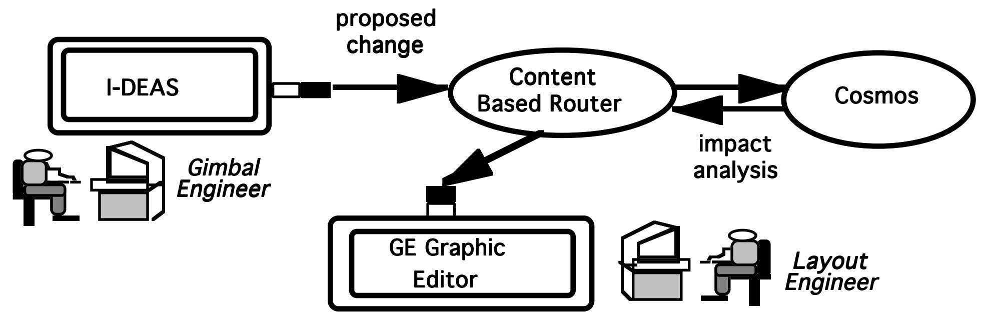


Figure 8. Step 4: Gimbal engineer actually proposes change

The gimbal engineer is satisfied that the second alternative is a viable option, and actually proposes it. Cosmos and the content-based router provide the scope of impact to all stake holders.

In this example, the only other stake holders are the layout engineers (see Figure 8). Given the scope of impact visualization in Figure 3, the layout engineers can quickly determine the negotiation variables: they have control over the strut inside diameter, strut outside diameter, and strut length. If the maximum strut stress is violated, they can propose a change to one or more of these engineering elements, or simply resist the proposed change. In any case, negotiation with respect to this scope of impact between the gimbal engineer and the layout engineers will continue until a resolution is reached, with

Cosmos dynamically updating the scope of impact visualizations to reflect proposed changes. In Figure 9, the layout engineers have decided to change the inside diameter of strut1, which will allow the strut to handle the somewhat higher gimbal mass within acceptable stress limits. Cosmos distributes this new information to all stake holders: the scope of impact that they all share involves two linked change proposals.

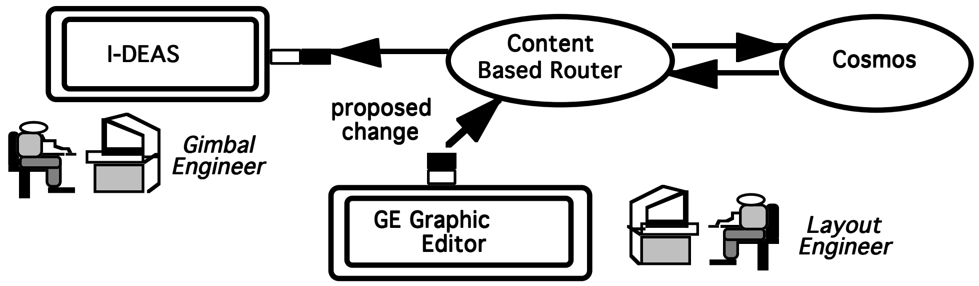


Figure 9. Step 5: Change is conditionally accepted

The layout engineer examines the proposed change via the scope of impact and proposes a change in his part of the design that will accommodate the proposed change. Cosmos and the content-based router inform all stake holders of the new scope of impact with the now linked change proposals.

In this case, the layout engineers' proposal is acceptable to the gimbal engineer, the only other stake holder, and the negotiation is complete. In general, if one or more of the stakeholders were not satisfied with this change, the negotiation would continue. Furthermore, new stake holders might be brought in if proposed changes start affecting other parts of the design. The complexity of engineering negotiation is inherent: Cosmos provides structure, automatic discovery and inclusion of stake holders, and automatic sharing of negotiation context.

5. Reactions

The Phase I Cosmos implementation has been demonstrated to a wide audience. The feedback from the engineering community has been quite positive with respect to the key elements of the Cosmos approach: engineers being able to experiment with alternatives in private before formally proposing the change; responsibility for involving stake holders being undertaken by the computational environment; and automatic sharing of visualized negotiation context being incorporated into the everyday engineering process. Large project engineers who have seen Cosmos have emphasized the role that higher level system engineers take during engineering change negotiations. In this regard, the scope of impact visualization can also provide context for the system engineers to make informed decisions. In fact, the scope of impact visualization can serve as the major form of communication between component engineers and system engineers, supplementing (or, better, replacing) formal engineering change notice procedures.

Cosmos has also been of interest to the part of the computer science community that is addressing the integration of heterogeneous software tools in distributed network environments. In particular, Cosmos has provided an opportunity to test knowledge level integration concepts such as KIF, KQML, and content-based routing. Our finding has been that Cosmos fits in well as a mediator in the computational infrastructure. The message-based interaction paradigm, the KIF/KQML languages, and the facilities of the content-based router have proven adequate for initial Cosmos needs, though the Cosmos Phase I implementation led to several refinements and enhancements of this infrastructure technology.

Finally, Cosmos Phase I has shown the need to expand and improve our representation of engineering knowledge and the reasoning mechanisms that work on it. For example, in the Phase I implementation we restricted our representation of constraints to allow straightforward modeling of MACE engineering knowledge with existing technology. It is clear from our experience and the demands of working engineers that we will have to provide a more in-depth constraint representation and reasoning capability. Also, our current ontologies and model cover only a small portion of the total MACE design. These must be significantly expanded in order to test the Cosmos concept at real engineering scale.

We are using our experience with creating and demonstrating the Cosmos Phase I implementation to inform the design of our Phase II system.

6. Cosmos Phase II

For Cosmos Phase II we are focusing on demonstrating that the Cosmos concept will work at the scale required to support large engineering projects. The major areas of advance over Phase I will include: more powerful commitment reasoning, more well-founded ontologies, more informative visualizations of the scope of impact, and a higher bandwidth communication channel among engineering change proposers and stake holders.

A key area under current investigation is the use of a more powerful commitment reasoning strategy. Our underlying knowledge representation system Loom gives us access to a powerful classification mechanism as well as to more standard production rules and other reasoning mechanisms. For the Phase I system, all commitments were represented in terms of production rules due to limitations in the Loom's constraint representation and reasoning capabilities. While production rules were adequate for Phase I, it is evident to us that the approach will not scale up to the thousands of constraints that must be represented to capture realistically large engineering models. For Phase II, we are working with the Loom implementers on a declarative representation of commitments that can take advantage of the Loom classifier to propagate constraints and detect violations.

Another area requiring considerable advance to enable scale-up is the use of externally produced ontologies. The Cosmos project is not intended to be a producer of engineering ontologies, but rather a user of engineering ontologies. For Phase I, we collaborated minimally with ontology researchers to develop an ontology of MACE concepts and their interrelationships. For Phase II, we will be exploiting other engineering ontologies to a greater extent. For example, the engineering math ontology "units-and-dimensions", developed under the SHADE project [SHADE], is a likely candidate. Cosmos Phase II will in fact be a good test of the adequacy of the evolving technology of "portable ontologies" (see [Gruber]).

We also plan to improve the scope of impact visualizations that are presented to engineers. The current visualization presents a tree of commitments with each leaf node itself potentially being expandable into a sub-tree. While this visualization presents much of the information that engineers need, feedback from the Phase I implementation has shown that it is not adequate. First, engineers have proposed presenting a visualization of the *degree* of violation; e.g., telling the engineer not just that the stress on strut1 now exceeds the maximum allowable, but showing the engineer a visualization of how much the stress is over the limit. A second visualization extension involves merging or simultaneously displaying more than one scope of impact pages. Since these pages are generated from a common engineering change, there is often a significant degree of common information. If we can determine perspicuous visualization of this multi-dimensional information, engineers will be able to much more rapidly assess the impact of a change in a large highly interdependent design. Also, we have some human factors work to do in terms of shapes, colors, and graph drawing algorithms.

A final important area under current investigation is increasing the bandwidth of interaction among engineers negotiating over a scope of impact. In the Phase I system, an engineer would propose a change that would be forwarded to all other stake holders. The stake holders have a limited set of actions open to them to respond to the proposed change: they can accept the change, reject the change, or accept the change contingent upon an additional engineering modification taking place. Further, while the original engineer views potentially several different scopes of impact in response to several alternative changes, the stake holders only view the scope of impact selected by the original engineer as the "best". For Phase II, this shared negotiation context will be expanded significantly to include all alternatives. One important motivator for this expanded scope of impact implementation is that on large projects, it is very difficult for negotiation to occur at the same time for all stake holders. Once an engineer forwards scope of impact information to the stake holders, those other engineers should be able to navigate the scope of impact information, experiment with changes that must be made in addition to the original change, and receive Cosmos impact analysis information on the new changes. All of this interaction must take place in a shared environment -- that is, engineers must have navigation-level access to the alternatives and rationale of their fellow stake holders in the negotiation.

7. References

[Baudin] C. Baudin, *et al.*, "Recovering Rationale for Design Changes: A Knowledge-Based Approach", *Proceedings of the IEEE Design Capture Workshop*, Los Angeles, 1990.

[Forbus] K. Forbus, "The Qualitative Process Engine", in *Qualitative Reasoning about Physical Systems*, D.S. Weld and J. deKleer (eds.), Morgan Kaufmann, San Mateo, CA, 1990.

[Gruber] T. Gruber, "A Translation Approach to Portable Ontology Specifications", *Knowledge Acquisition*, Vol. 5, No. 2, Pp. 199 - 220.

[Grudin] J. Grudin, "Groupware and Coopertaive Work: Problems and Perspectives" in B. Laurel (ed.), *The Art of Human-Computer Interface Design*, Addison-Wesley, 1990.

[KIF] M. Genesereth and R. Fikes, *Knowledge Interchange Format Version 3.0 Reference Manual*, Computer Science Department, Stanford University, Technical Report Logic-92-1, 1992.

[KQML] T. Finin, *et al.*, *Specification of the KQML Agent Communication Language* Enterprise Integration Technologies Technical Report 92-04, 1992.

[Mark 92] W. Mark, *et al.*, "Commitment-Based Software Development", *IEEE Transactions on Software Engineering*, Oct, 1992.

[Mark 93] W. Mark, "Shared Dynamic Negotiation Contexts", in *Proceedings of JJCAI Workshop on Conflict Management*, International Joint Conference on Artificial Intelligence, Chambery, France, 1993.

[PACT] M. Cutkosky, *et al.*, "PACT: An Experiment in Integrating Concurrent Engineering Systems", in *IEEE Computer*, Vol. 26, No. 1, January, 1993, pp. 28-38.

[Petrie] C. Petrie, "A Minimalist Model for Coordination" in C. Petrie (ed.), *Enterprise Modeling*, MIT Press, 1994.

[SHADE] J. McGuire, *et al.*, "SHADE: Technology for Knowledge-Based Collaborative Engineering", in *Concurrent Engineering: Research and Applications*, Vol. 1, No. 3, September, 1993, pp. 137-146.

[SHARE] G. Toye, *et al.,SHARE: A Methodology and Environment for Collaborative Product Development*, Stanford Center for Design Research Technical Report 1993-0420, 1993.

[Wiederhold] G. Wiederhold, "Mediators in the Architecture of Future Information Systems",*IEEE Computer*, Vol. 25, No. 3, Mar, 1992, pp. 38-49.