High Level Collaboration Using Knowledge Structures

Abstract

A method of collaboration where collaborators contribute pieces of undirected active structure is described. Such pieces can be easily integrated into a larger structure, as any piece is undirected, and can be linked either directly or indirectly to other contributions. The structure is also self-modifiable, so a piece of structure can carry with it substructure that allows it to automatically link with the other structure it finds. Contributions are not just take it or leave it, as pushback from the larger structure occurs, making it the responsibility of all the collaborators to ensure that a consistent and coherent structure results from the contributions, each of which must be consistent in itself. Several areas, including requirements elicitation and collaborative design, are used as examples of collaborative decision-making, requirements at a conceptual level coalescing into a consensual design, with the initial conceptual structure surviving and becoming more detailed without a break occurring in the methodology of collaboration.

Key Words

Active structure, relations on relations, existential control, undirected stitching, self-modifiable.

Introduction

We will assume that collaboration implies contribution by the collaborators of structures that represent knowledge. We further assume that the architecture of the DSS should be largely determined by the contributions, instead of the contributions having to fit within a predetermined architecture. If so, the structure needs to be of a special form, a form that is self-merging with other contributions – an active form. The merging of passive contributions requires a much greater level of sophistication in the merging entity. The entity would need to "understand" each of the contributions, then merge its understandings, and in the limit, no entity can be expected to understand all the contributions. Collaboration in decision making also implies that each collaborator can see and understand enough of the decision making machinery and the context to have confidence in the decisions being made, and preferably can see how what they do contributes to the decisions made.

Two models of collaboration become possible – one model is that of people sitting around a table and contributing ideas until a consensus is reached. This works, but only for either relatively shallow decisions, or decisions which cannot really be solved collaboratively, the complete solution needing to exist first in one person's head. The other model is that of contributions building up from a foundation until the peak decisions are reached. Again we need to break into two models – whether people contribute bricks to a heap, without any sense of what is being built, or there is so strong an architecture that the decision is already made, irrespective of the contributions, because it is already determined how they will be stacked. Neither model seems satisfactory. This is the classic dilemma of either bottom up or top down decision making – how to have sufficient cast so the bricks are useful, but not so much that they already have a predetermined shape and only fit one way. We will be suggesting that making the contributions undirected, to give them the properties of active knowledge, is one way of achieving effective collaboration. We will also assume that slabs of free text are not sufficiently integrable to use as contributions, due to difficulties of connection to the concepts internal to the contributions, and ambiguity. Instead we will describe active structure as a suitable basis.

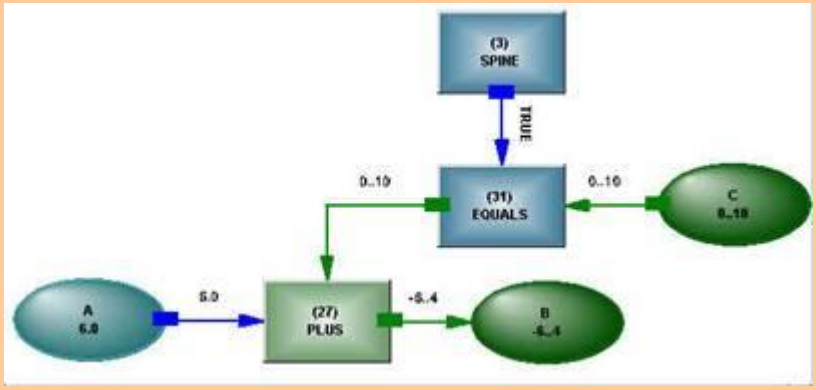
Active structure comes in two forms – one a simple form for low level problems, and a complex form for more complex problems. We will briefly describe the simple form, then proceed with the more complex form, as it is better suited to the support of human collaboration in decision making.

Simple Active Structure

Active structure is structure made up of nodes, operators, links. All dynamic information exists in the links, or in structures accessible from the links. Nodes maintain conformity of state in all their links, operators use changing information in their links to change information in their links, and links maintain information and propagate it throughout the structure. The links in the structure are notionally undirected, with information flowing in whichever direction is appropriate for the semantics of the operator. Propagation of a change of state in a link indicates that the node or operator to which it is connected should be given control, so change of state within the structure controls phasing of activity within the structure. This relatively simple formalism holds for simple relations like

A + B = C

where the operators are the plus and the equals, and the nodes are the variables, A, B, C and the head of the spine (not explicit in equation, but the equation is written on a logical surface, which the spine represents). Figure 1 shows a diagrammatic representation of the structure (the diagram is a direct representation of the machinery, not just a conceptual map for us to understand the operation of an algorithm). In this case the operators maintain consistency on their links according to their simple semantics (a PLUS operator may be fed its connections by a dynamic list, and with ranges in every one of a hundred links, any one of which can be an input or an output, it can get rather complicated), and the variables maintain conformity. The structure shows the link between logic and numbers at the EQUALS operator, and the structure is propagating ranges of numbers in a direction determined by the logical state of the incoming information (the directions shown by the arrows in the diagram are dynamic, the structure itself represents knowledge about a relation among objects and is undirected).



This simple formalism works well for numbers, lists and logic, and structure conceptualised as a logical object, but is not adequate if we wish to handle the more complex areas of decision support.

We will introduce the extensions of complex active structure that are relevant for collaborative decision support systems (more detailed examination is available in [2]). The extensions have proved necessary for processing scientific and legal text, with the implication they are also necessary for other complex forms of human collaboration in these areas, forms that are at the same level of complexity as collaboration through speech.

Detailed Structural Existence

In the simple form of active structure, the existence of an equation like A + B = C or an inequation like D + E < F can be controlled at the level of the statement by controlling its connection to the logical spine. Where there are complex and layered relations – “the option to extend the lease” – existential control needs to be exercised over the individual relations – each relation needs both a

logical and an existential connection. This isn't just to represent the structures and states implicit in free text, but also to more precisely describe the basis for a decision. The option may have expired, the extension only available in increments, or the lease terminated for breach – relations like these carry dense information in multiple dimensions. We can have relations with both logical and existential connections, but there is a strong relation between the two connections – a relation can't be true if it does not exist, and it can't be logically truer than its existence value – "If I can pay you, I will". You can see what happened here – we took the existential truth value of the relation and used it as the antecedent, with the consequent the logical truth value. The relation operator maintains the necessary relation between its two truth states, just as the PLUS did for its connections in the simple form. Do we really need this distinction? –

I can jump puddles.

I will pay you next week.

We use the distinction all the time in communicating with other people, or reasoning about their actions and intentions, certainly we should expect it in contributions to a DSS.

All of the logical connectives (AND, OR, IF) need to be able to accept any mixture of existential and logical states, because we as humans cannot be bothered doing something so obvious (this gets to be important across contribution boundaries).

Temporal Structure

If the system has control over existence, then it becomes important to be able to represent when an object or relation was created, when it was destroyed, how long it may survive. Most relations inherit durations, while some relations instantaneously create and destroy other relations – "He started working at the bank last week", "the bridge was destroyed by fire". Figure 2 shows the temporal structure of an "occupy" relation (top left corner - a relation can inherit a time period, the time period can be inferred from connected relations, or it can have an explicit component – the COMPONENTS operator - of Time Period, which inherits or has attributes of Start Date, Finish Date and Duration).

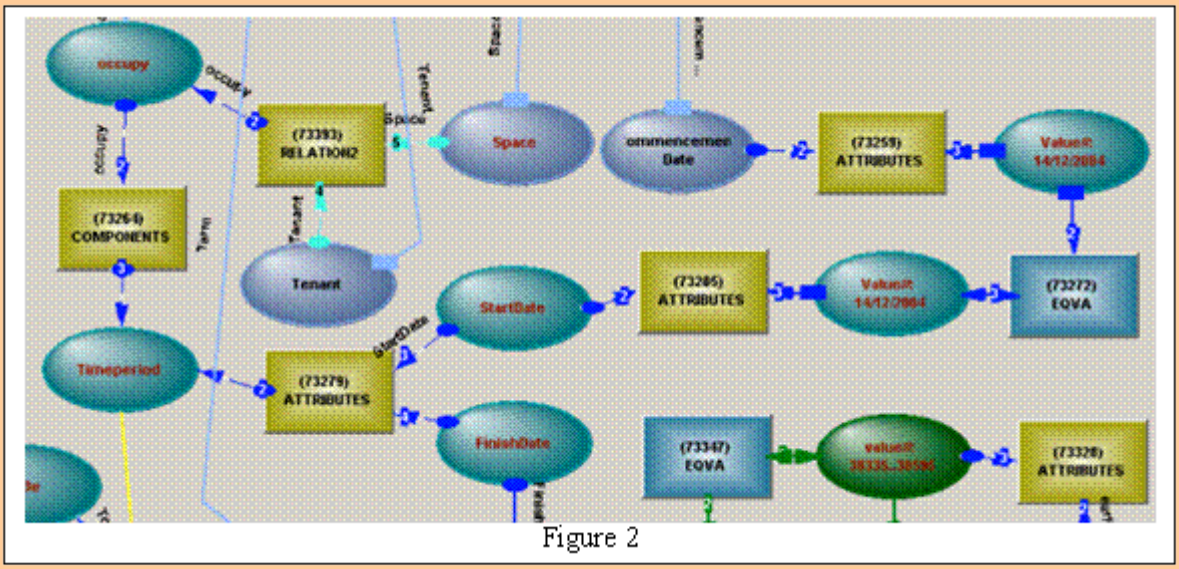


Figure 2

Time is handled using diffuse operators – operators which do not exist at the particular level until they are required, and then hunt out objects to which they should connect. Layering is handled by dynamic construction. That is, if an operator

has constructed itself and needs input, an operator is constructed to provide that, and so on. This avoids the problem of multiple parallel activity, which a structure driven by a single processor cannot perform.

Relations as Objects

The notion that objects and relations should be treated as different things comes from the evolution of the concept of objects from programming languages, with their strong dichotomy between locations and operations. In the real world, the one we talk about, we know that a lease is a relation among a lessor, a lessee and property, a car is an assembly relation on its components, a process is the doing of something. If we allow relations to be objects in our formalism, then we have a much more coherent and general representation of the world. Relations acquire properties by inheritance in the same way as other objects, but they also have parameters, which may be relations.

Figure 3 shows a relation, ToSell/ToBuy, and its parameters, or the things it relates. The relation operator has a head object, which allows the relation to be an instance of a base relation, to inherit properties and be connected as a parameter to other relations to form chains of relations. The relation also has explicit existential and logical connections, things a simpler form of relation such as the PLUS operator lacks. Non-arithmetic relations in general require a specific existential connection – a contribution to a DSS may want to signal the difference between "the buyer cannot..." and "the buyer did not..." just as much as we need to do in speech. Joining buyer and seller together with the object being sold in the one transaction relation should make it clear how a more complex and general representation of a relation can reduce the length of inferential chains, and represent many different nuances on the one transaction, using temporal control over the existential and logical connections –

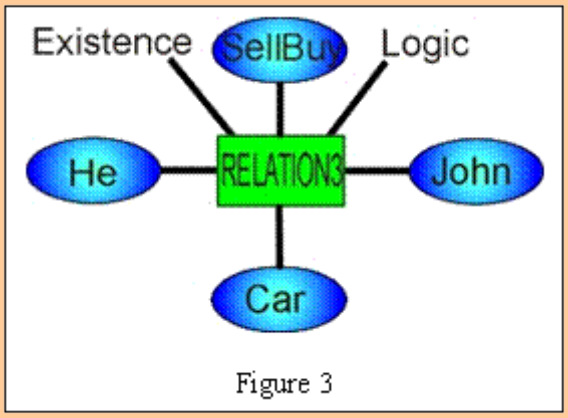


Figure 3

He sold/has sold/is selling/will sell/may sell a car to John

He did not/could not/would not/should not sell a car to John

John bought/is buying/may buy a car from him.

Object Groups

We can give any grouping of objects properties through inheritance or through connection with a relation, properties which are then inherited by the members of the group – "the team won" – the members are winners too. This works both ways – if we can show that every member of the group has some property, and the group can have that property, then the object group acquires that property. This gives us a form of computable inheritance from realised classes.

Computable Inheritance

Treating relations as objects and allowing them to acquire properties through inheritance greatly increases the density of the inheritance structure. We cannot assume the inheritance structure is static or known a priori. We should expect collaborative contributions to change the inheritance structure, requiring it to be dynamic. Frequently, disambiguation will turn on changing the location of the connection of an object into the inheritance structure.

Two situations arise where inheritance is not fully known, even after contributions. One is where an object is part of another complex object, but we are unsure which part. "Part of the building is on fire". Until we know more about the "part", it needs to potentially inherit all the appropriate properties of the building and any (combustible) component and its container, and we need to interpose some control over the inheritance path, initially allowing alternatives and pruning them as we learn more. A related case is where we know an object is a member of a group, but we don't know which member. We are told a member of staff is to receive an award. That member has potentially acquired the individual properties of every staff member. We are then told the person is female, sloughing off many possible properties ("not the janitor"), and so on. Under some circumstances, inheritance needs to be controlled, does not just come from above in the hierarchy, and the methods of constraint reasoning can be used to aid in narrowing the possibilities.

Relations on Relations

We gave an example – "the option to extend the lease" – in this example, there are three relations, all in a row. If human intention is involved, a whole new layer of relations is added – "he thought he needed to exercise the option to extend the lease" (in this case, the statement is incomplete as it is the term of the lease that is extended, and it is assumed that the option exists, but it is not activated by this structure). With existential control on relations, and seeing relations as objects, relations on relations is easily handled. A relation on relation structure to any depth is built by connecting the relation objects and their parameters, which may be relation objects. In AI, we have struggled without this facility for a very long time.

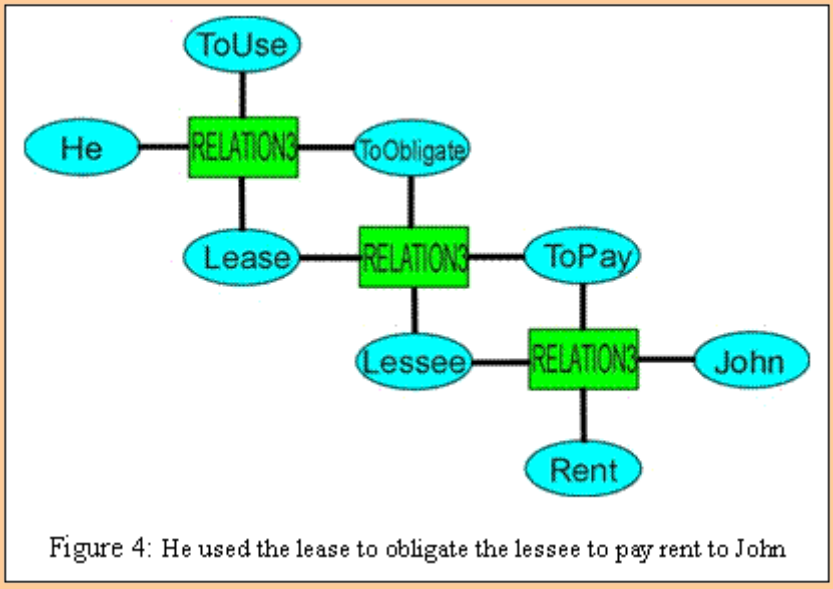


Figure 4: He used the lease to obligate the lessee to pay rent to John

Existential, logical and inheritance links are not shown in figure 4. For a complex DSS, there may be hundreds of thousands of elements required in

the active structure, many of which may change their state rarely. It means that all the context of the DSS application is continually able to be sensed (but while there is no change, there is no activity, so a large and densely connected structure is not necessarily wasteful).

Active Maps

We have been describing components, like relations, and how they may be joined to form larger structures. This is a reasonable analogy of what happens when a person builds cognitive structure starting from nothing, but people rarely do that, and collaborators never do. A more realistic and more complicated problem arises where we have significant pieces of knowledge – contributions that form relatively independent islands of knowledge - and we wish to combine them into a DSS, based on their interaction with each other. Humans have an ability we call the "semantic octopus". Something arrives at a point of disconnected activity, grasps the free connections, ties a knot, and moves away. It has about the same range as an octopus – somewhere between six and nine free connections. We haven't yet conceived of how we can directly match this, but if contributions are to be stitched together into a seamless whole, some similar mechanism is required. An active map connects itself to the things it finds, then runs a constraint reasoning problem involving itself and what it has connected to. If the problem solves, further structure is built and the active map detaches itself.

A Package

The ability to actively represent relations on relations relies on existential control of individual relations, on relations being conceptualised as objects, so they may be part of the inheritance hierarchy, and on relations inheriting a component of time period, which determines their existence. These properties of the structure are not divisible into separate properties which can be added individually – they support each other synergistically.

Some Relevant Issues

Assembly

It is much easier to assemble active structure than it is to assemble directed algorithmic components of an overall algorithm. It is also much easier to assemble active structure than some preconceived pieces for an application anchored in a particular domain, where one relies on the foresight of the programmer to predict all the interactions and operations that may occur. For many systems, most of the skill in creating a DSS consists in twisting algorithmic elements to achieve some end that was not foreseen. Pieces of active structure can be knitted together with other active structure to cross mismatched joins so the resulting structure is seamless, or they can come with hooks that seek out other pieces of structure and then dissolve after their work is done. Active structure can be built and tested as self-contained islands of knowledge, which later coalesce into archipelagoes and then continents, using the properties of self-modifiability and immediate activity on connection.

Common Understanding

Collaborators can achieve a common understanding by exercising parts of the structure, by observing how other contributions feed back into and affect their contributions, by observing weaknesses and fault lines. By being exercisable while incomplete, and allowing temporary structure to be placed on top, the structure lends itself to being a foundation for further work at each stage of its building. Sometimes the common understanding will break down as the

structure nears completion, or even after it has been in use for a while and does not provide the decisions some of the collaborators expected. It is then that the ability to isolate the faulty structure becomes critical, particularly if the faulty structure crosses contribution boundaries.

Complexity

Complexity comes in two forms. One is where the complexity lies wholly within the bounds of the contribution, and while needing connections into other areas, the complexity is managed by one of the collaborators. The other form is where complexity emerges from the interaction of different, relatively simple, contributions, with no collaborator expecting it or responsible for managing it. Emergence of complex and unexpected behaviour is typical of many systems, so it becomes important to ensure that the tools of systems analysis are available. Typically these include localised or controlled activation, tracking of behaviour with different, including extremal, inputs, hypothesising and testing, temporary removal of feedback paths. As active structure allows probing at an atomic level of activity, fine control over existential and logical states, hypothesising, and the addition or removal of structure at any point, it is ideal for the kinds of analysis required at a system-wide level.

Data collection and Exchange

We would argue that any data comes with a structure and a process, and a model of the process can be easily built in active structure. While statistics can be used to seemingly provide information from data without a process being known, this only works at a shallow level. It is just as easy to produce a model component, which, for given inputs, produces a desired output, and vice versa. That is, the data model is undirected, and while the transformation from output to input would usually be one to many, it follows the undirectedness paradigm of the rest of the structure, and the DSS is easier to understand and debug as a result.

Utilising Context in Knowledge Management

There are two forms of context in a knowledge-based system. There is the overall context in which the DSS is embedded, and there is the context of each decision, which is not predictable. The first form of context can be represented by a very large amount of structure which has a low probability of change. If any part of it does change, that change will flow through the system, causing it to alter its behaviour. The second form of context can be passed from one decision point to another by creating and propagating an active structure, allowing subtle description of the transient context.

Examples

Planning and Scheduling

There is a continuum in planning and scheduling of projects, from the initial concept, through tendering to program and project management. We have the people who conceived of the need, the people who let the contract, the people who manage the project, and the people who will own the project product.

Figure 5: Planning spectrum

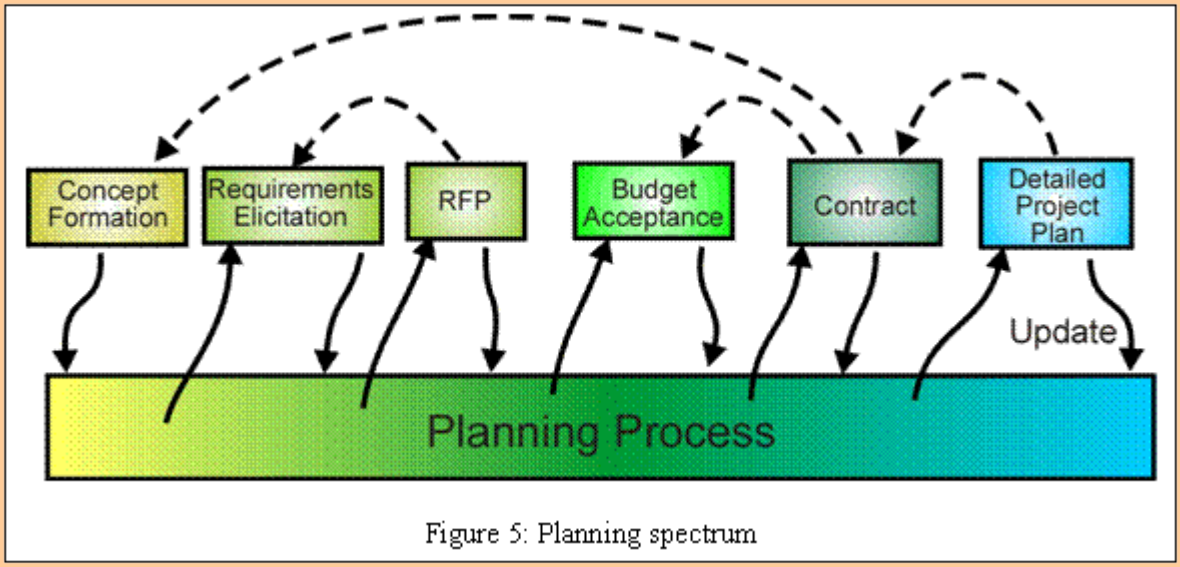


Figure 5: Planning spectrum

The planning process may look like an orderly progression through time, but the project can run into problems, the contract price blows out or there are delays, calling the concept into question. It is normal for back-connections to play a significant part in the process. The current decision support tools are algorithm-driven and the process is directed and segmented, so that people who could collaborate are held at arm's length by a methodology that cannot allow their input until it becomes necessary, or their input becomes unconnected from the current project plan by the passage of time. It is much more powerful to use a broad spectrum planning approach, which, with the same methodology, supports the concept development, the cost-benefit analysis, the tendering process, requirements management, the contract management, the project management, the owner's view. Active structure allows every stakeholder to contribute their requirements and constraints to the DSS, so that the DSS continuously supports a model which includes the what, the why, the when and the how much, rather than one aspect. There is a stitching of the contributions that is required, shown in figure 6, as the contributions will usually not overlap. The stitching not only needs to be undirected, as influences may flow in any direction, but must be able to represent any process, which is why we describe the structure as active. Only if the stitching is capable of connecting anything to anything, through whatever intermediate structure is necessary, can synthesis be achieved outside the head of one person. Otherwise, no matter how diverse and detailed the contributions, a person must maintain synthesis, which is impossible for large and complex projects over a long time scale.

If the project drifts off spec, the contribution from the strategists drags it back (because the initial specifications, the why, are part of the structure), if the project drifts from what the owners can pay, it will be killed by the owners' contribution to the model (and the project manager can feel the potential pain of failure, and plot another course).

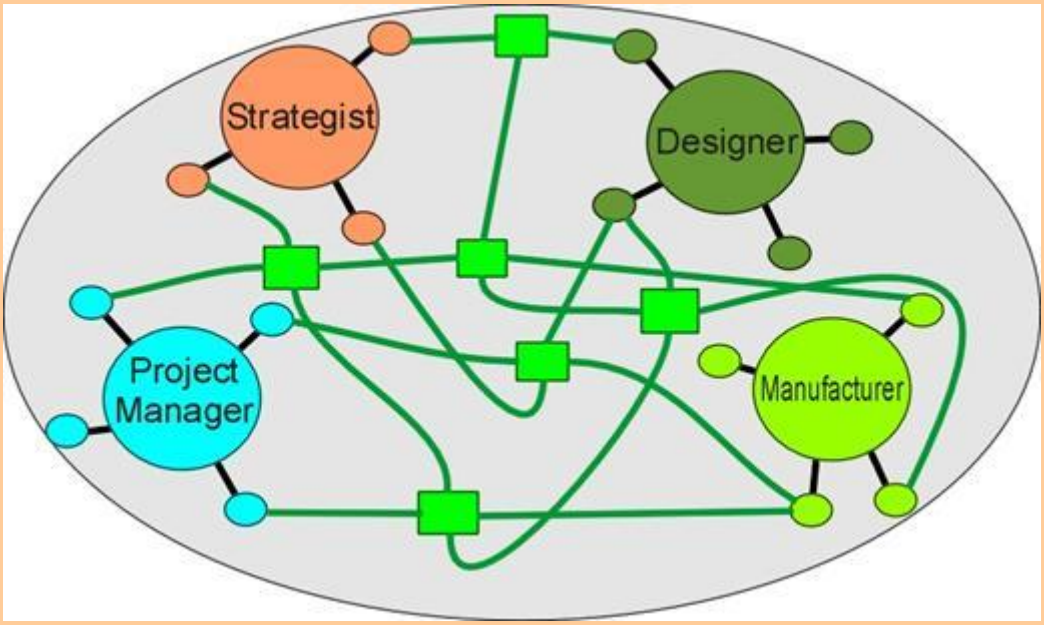


Figure 6 - Stitching

Figure 7 shows a simple form of an ACTIVITY, with only logical control from a spine or logical condition (costs and resources associated with the activity are not shown). A more complex form makes it the same as any other relation, with the same subtlety of start and finish date.

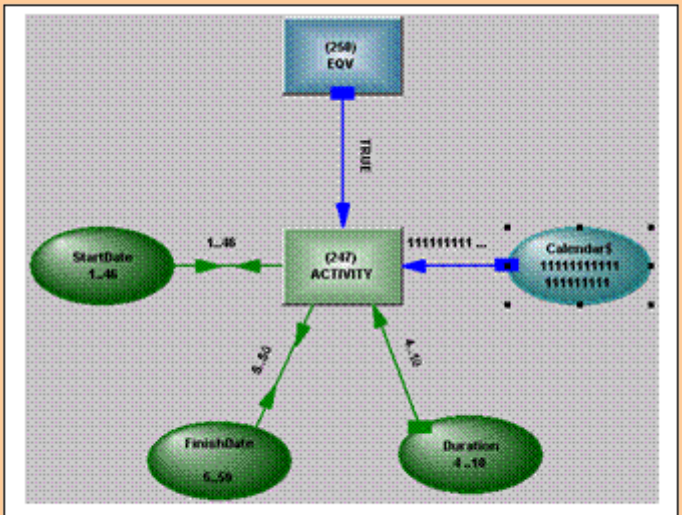


Figure 7: ACTIVITY Operator

People may be concerned at the cost of modelling projects at this level of detail, but the cost of large failed infrastructure projects regularly runs into billions of dollars, so providing the capability for all the stakeholders to directly collaborate to the success of the project is a small price to pay.

Requirements Elicitation

Requirements elicitation is only a small part of the planning spectrum, but it carries its own problems of collaboration. The ideas will usually be woolly, and there may be several possible scenarios, each with its own particular details. Trying to put a hard edge on everything too early is a mistake, but carrying forward flawed concepts longer than necessary is equally a mistake, as attributes of an impossible choice may seem attractive and spawn further investigation down dead ends, or people will defend worthless concepts because they have invested their time in them. Active structure can be used to outline the possibilities and slowly draw in the constraints, switching properties on and off as scenarios are examined. What emerges is an outline, but with sufficient connection to reality that a design is possible.

Supply Chain Management

Supply chain management has the need for people to collaborate across many domains – scheduling, price, quality assurance, manufacturing. If we broaden the definition, and include the distribution of the good produced by the supply chain, we bring in other groups, such as sales, management accounting, who may wish to collaborate if their direct involvement could be supported. One aspect of supply chain management that is particularly difficult is the supply contracts, which are usually handled as repositories of opaque knowledge until something goes wrong (it usually has to go badly wrong, at that). A collaboration occurred in creating each contract (often with little connection between contracts or any sense of being part of a supply network), and the sense of this connection is largely lost in operation. Using active structure permits the structure of each contract to be extracted, so the collaboration it represents can be actively represented. The drivers of the contributions in each of the different domains are quite different, making it preferable to use a methodology that turns all the contributions into an integrated whole and allows tradeoffs backwards and forwards into any domain. Within a domain, cost of components needs to be weighed against cost of assembly, and then outside it, cost of delivery of the assembled product. An initially undirected structure assembled from contributions would seem an appropriate choice, both within each domain and over all the domains involved.

Knowledge-Intensive Collaborative Design

It sounds trite to suggest that the collaborators contribute knowledge, which is assembled to produce the design, together with the structures that support the design, such as market surveys, product standards, even aesthetics. It requires that their contributions be in the form of knowledge that is active and can be assembled without limits imposed by preconceptions of what is possible. If the design must fit within a predefined framework that suits horses and carts, the emergence of a design for a helicopter is unlikely.

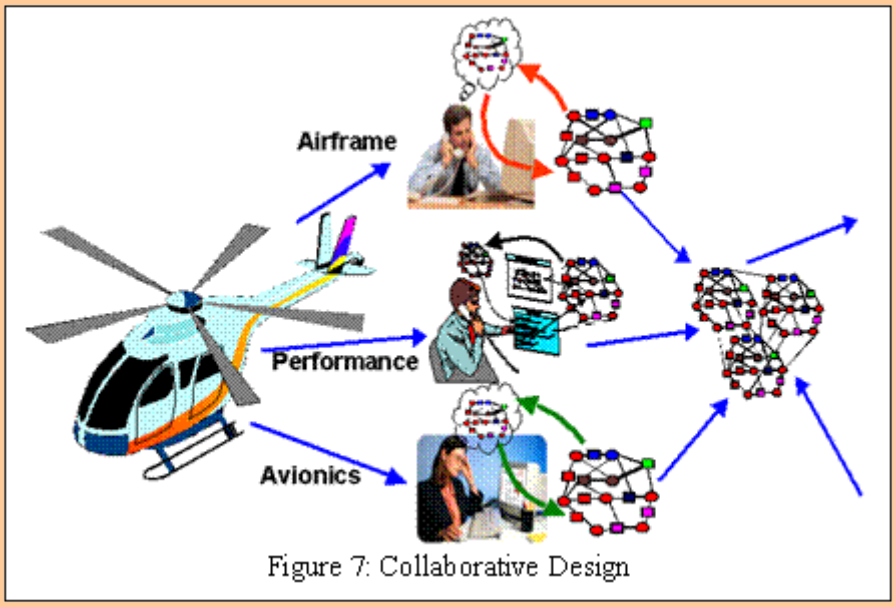


Figure 7: Collaborative Design

This is a crunch test for collaborative design – if the framework of the collaboration cannot support new ideas, ideas that were not envisioned when the framework was created and which would require either a subtly or radically different form of knowledge to be used, it is not design at all, but merely "tiling the plane" of what is possible within the framework. Active structure would seem an appropriate form to allow new designs to emerge. Will the collaborators need to consider the existence of the different components of the design – undoubtedly. Will the analysis of the quality and likely success of the design use relations on relations – probably. If the methodology they use cannot support these things, the collaboration process is likely to be heavily compromised.

Conclusion

We have described a methodology which is capable of representing and activating knowledge, allowing collaboration at the level of knowledge. The methodology supports a close integration of propositional, existential and temporal logic, and relations among objects. This integration frees the collaborators from concern about process, and allows their individual contributions to contain ambiguity, which can be washed out by activation of the structure that is built from the contributions. An active structure, like knowledge, is undirected, so that collaboration results in the creation of a much more powerful system than one which is geared towards a particular end before it receives contributions. Emergence of unexpected behaviour can be handled by allowing the system to analyse itself for consistency and coherence. Building active structure allows collaborators to contribute at the level of complexity attainable if they wrote or spoke to each other, but with easier integration, less ambiguity and more rigour, as each contribution can be tested for consistency with the structure that has already been built. We emphasise the level of complexity – active structure can be very dense in its possible connections, as dense as is required to reflect the complexity – no more, no less.

Bibliography

Whither Directionality (http://www.inteng.com.au/whither.htm)

Active Structure Extensions
(http://www.inteng.com.au/NLP%20Operation/extension_to_active_structure_to.htm)

Cognitive core of an EIS (http://www.inteng.com.au/cognitive_eis.htm)