Contextual Assistance in User Interfaces to Complex, Time-Critical Systems: The Intelligent Zoom

Lyn Bartram, Russell Ovans, John Dill, Michael Dyck, Albert Ho and William S. Havens

Centre for Systems Science
Simon Fraser University
Burnaby, BC V5A 1S6
lyn@cs.sfu.ca

Abstract

Network supervisory control systems (such as those used in telecommunications) are characterised by very large information spaces, highly concurrent activity and time-critical response requirements. The user interfaces are typically so complex that the task of manipulating the interface interferes with the tasks of managing the underlying system. This paper presents the **intelligent zoom**, a technique based on a fisheye-lens paradigm that combines dynamic representation aiding with a distorted-view algorithm which permits multiple "focus points" while maintaining the overall network context. The merge of adaptive data presentation, "detail-in-context" views and direct manipulation provides a more seamless path through the information space, and is well suited to working with nested levels of detail in network displays. A distributed reasoning architecture based on a multi-threaded programming style ensures efficient performance.

Résumé

Les systèmes de contrôle supervisoires de réseau sont characterisés par de tres grands éspaces d'information et un grand nombre d'activités concurrentes. En plus, l'opérateur est obligé de réagir aux événéments sans délai dans les périodes criticales. Les interfaces sont souvent si complexes que la navigation et la manipulation de l'éspace d'information nuisent à la tâche principale - la gérance du système de contrôle et du procès. Cette oeuvre présente le **zoom intélligent**, une technique basé sur une déformation "fish-eye" qui combine une présentation adaptive avec un algorithme pour créer des vues déformées qui permet la visualisation de plusieurs points d'interêt même en maintenant le contexte du réseau. L'ensemble d'une présentation adaptive des données, une vue de détails en contexte, et la manipulation directe donne une réprésentation du domaine dans laquelle il est plus facile à naviguer, et qui est bien adaptée a l'interprétation de plusieurs niveaus de détails hierarchiques.

Keywords: graphical user interfaces, adaptive data presentation, graph visualisation

Introduction

Supervisory control systems (henceforth **SCS**), such as those used in network management, are characterised by very large information spaces, highly concurrent activity and time-critical operator and system response requirements. The extent of these systems has grown enormously in recent history, and the role of the operator has evolved from low-level manual control to a high-level supervisory function. However, the user interfaces typically still reflect the physical process at a low level of detail. Such "one sensor, one display" interfaces[11] are overwhelmingly complex; operators are often faced with using hundreds of displays and thousands of data points in their tasks of monitoring and controlling the physical system.

Problems arise because the task of managing the interface (finding the information, interpreting it and carrying out the controls in a timely manner) imposes significant cognitive overhead and impedes the task the interface is supposed to serve: managing the control system and physical process.

Our goal is to provide support for the operator by off-loading some interface management, freeing the operator to concentrate on his primary function. A significant part of the operator's interface manipulation task involves amassing suitable information to provide proper working *context*. One way of reducing interface-imposed overhead is to convey information to the user in a context-dependent manner: that is, dynamically configuring the display to present information that is consistent with current and previous status (alarm) messages from the SCS and current and previous user manipulations of the interface. We take a bifurcate approach to building a context-dependent network display by (a) providing detailed views within the overall network view (maintaining both local and global context), and (b) dynamically determining the most appropriate representation for the data at the current detailed area of interest given the current system and interface state.

This paper describes the **intelligent zoom (IZ)**, an interactive display technique that utilizes adaptive data presentation within a "fisheye view" of the overall network to provide a dynamic, context-dependent network display. Fisheye views (such as those reported in [5][12] [15][16]) show local detail in full while displaying successively less detail further from the area of attention and have proven effective in easing the "lost in space" problem associated with large information spaces[16]. The adaptive presentation algorithm is distributed across special reasoning agents. While the rules in each agent are very simple, they combine to produce the type of complex behaviour which normally requires a sophisticated and complete expert system.

Motivation

Our main goal is the investigation and development of techniques which produce context-sensitive interface behaviour without imposing extra overhead upon the user. The following research issues comprise the focus for this work:

- Coherent representation of large information spaces: how to organize the information space to be both usable and navigable by the user, while maximizing the use of limited display resources.
- Adaptive presentation: the problem of deciding display content based on information available at the time the display is produced; how to reconfigure the presentation methods on-line as operational conditions change and as the operator manipulates the user interface. (This is also termed *intelligent presentation* [13].)
- Dialogue management: how to best provide assistance in the management of communication between operator and control system in an environment of multiple, interrupting tasks.

Problem Statement

The task modelled is that of presenting, navigating, and interacting with a network of connected nodes, much like that encountered in network-based control systems found in the telecommunications and utility domains. Displays in such systems are usually organized as a set of fixed viewing configurations: multiple views of data are presented in different displays, such that it is impossible to simultaneously view a trend graph of one node with a schematic of another in the context of a unified display[6]. We want the interface to choose the best view of the network given the current operating context of alarms and user manipulations. Furthermore, representations should be correlated to alarm type, e.g., a *door ajar* alarm suggests a live video representation rather than a device schematic (an attempt to infer, in some sense, what the SCS is trying to say.)

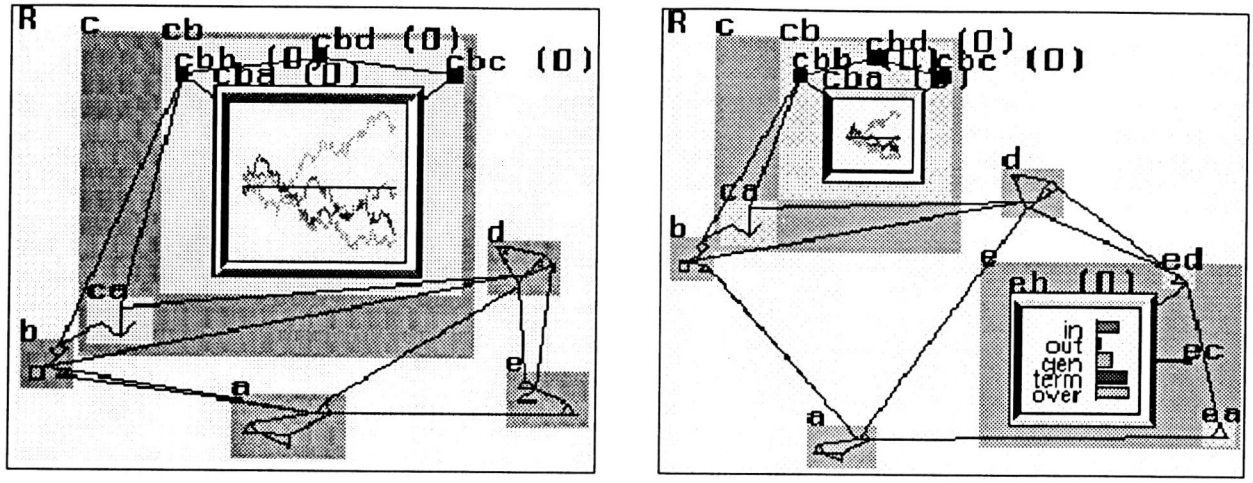
We treat context-dependent presentation in a large information space as a problem of managing limited operator(user) and system resources [13][14]. Operator resources are perceptual and cognitive modalities: the human cannot interpret the entire data space at once. System resources are the physical aspects of the interface, the most obvious of which is screen space: not all system data can be portrayed simultaneously. The demands for these limited resources must be mediated in an appropriate and timely manner, indicating a need for active reasoning. But the notoriously slow performance of the reasoning architectures used in standard expert systems (ES) renders them unsuitable candidates for SCS interfaces: clearly, an alternative approach must be used.

The Intelligent Zoom

The **IZ** combines the **continuous zoom (CZ)** algorithm with dynamic, adaptive representation choices for data visualization. The continuous zoom is a distorted-view technique suited to viewing of nested levels of details in hierarchically-organized network displays. It evolved from the variable zoom reported in [16] and extends the fisheye paradigm of [5] and [15] by supporting multiple detailed views, or *focus points*. It manages the display space by allocating more space to certain areas (**nodes** in the displayed network diagram) at the expense of others without removing or occluding the containing context. However, it is insensitive to the *contents* of the nodes: it is merely responsible for determining size and placement of nodes within some thresholds. In the taxonomy described above, we consider the zoom algorithm to be a knowledge source which manages and renders the *display space resource.*

In the CZ, the user views and navigates through a hierarchically-structured network of nodes and links, where nodes are of type either **cluster** or **leaf**. Cluster nodes are top- and intermediate-level in the hierarchy and contain clusters of children, who are in turn cluster or leaf nodes. Leaf nodes provide access to system data in the actual physical or logical points in the network. Nodes can be opened (showing the contents) or closed. The size of a node can be altered "continuously": that is, the user can expand and shrink it smoothly without opening or closing it. The size of a node depends on the size of its siblings and the size of its parent. The size of a parent is quasi-independent of the size of the children. The user makes a node *n* larger by:

- opening it;



a b

FIGURE 1. Continuous zoom displays before (a) and after (b) a second focus point is selected

- zooming in on it;
- shrinking/closing its siblings;
- shrinking/closing the siblings of its parent; or
- zooming one of its children beyond the maximum size that child can be in the current size of the node, thus causing the parent (*n*) to grow to accommodate the child.

Whenever a node is resized, it affects the size of other nodes in the network display. A node grows by claiming space from other nodes, thus causing them to shrink; the inverse is of course also true. The space is reclaimed from the siblings of the node, until the node has grown to its maximum within the parent; then the parent is expanded at the cost of its siblings, and so on until the top level is reached. There is a limit on how small a node can become, imposing an implicit upper bound on how big a node can be (since a node cannot cause its siblings to become too small). This ensures that a detailed view of one node does not remove the overall context: there is always a "complete" view, which may consist of mixed hierarchical levels of detail (Figure 1).

While it takes only one user action, opening a node actually affects two aspects of the node's visual state: it becomes transparent, so that its contents are visible, and it becomes larger. The aspect ratio of a leaf node is fixed to that of its current representation; the aspect ratio of cluster nodes varies.

Our example network has 45 leaf-level nodes and 3 hierarchical layers using 17 intermediate cluster nodes. Leaf nodes are considered to be the actual locations of interest to the operator/user (in that they represent actual physical or logical points in the network which the operator monitors); cluster nodes provide a convenient abstraction and a source of summary information. In this first version we were only concerned with leaf-level representation aiding.

Seven representation types were arbitrarily chosen as possible visualisations of the information in the leaf nodes:

1. schematic (a diagram of the devices and layout of the particular node);
2. a bar chart of analog values;
3. a trend diagram (a *yt* plot);
4. video (currently a bitmap);
5. text;
6. icon; and
7. closed.

Each representation has a minimum resource requirement and is more appropriate for certain conditions than other representations. Examples of resources are hardware/software (e.g., there can only be one video display at a time); cognitive (i.e., the demands on the user's comprehension and perception when interpreting combinations of representations) and display (e.g., colour and spatial requirements).

Reasoning: The "Intelligent Agents"

Adaptive presentation (specifically, suggesting a representation for each open node that the user has not explicitly specified) is the task of special reasoning agents (henceforth **IA**s). The reasoning task is difficult because interface resources are limited. In order to effectively solve this problem an interface must provide explicit support for the allocation of limited interface resources and for *representation aiding*[19]; i.e., choosing a contextually appropriate representation for data. We have proposed intelligent mediation (IM) [13] as an architecture and methodology for context-sensitive pre-

sentation that encompasses both resource allocation and representation aiding.

IM identifies the allocation of limited interface and human cognitive resources as the key problem in interface design. Specifically, IM:

- enforces a local encapsulation of knowledge particular to resource allocation, with separate and specific classes of mediators for each limited resource;
- specifies behaviour akin to a "semantic spreadsheet", such that resources are redistributed in response to change, rather than reallocated from scratch (reasoning is on-line rather than batch); and
- provides a framework for representation aiding. We treat representations as a limited resource; IM permits a mechanism for choosing appropriate representations given current operating context.

In the intelligent zoom, leaf nodes must compete for a limited set of representations. During this competition, we prefer that the more important nodes receive the best choices. That is, the representations mediator should use the degree-of-interest (**DOI**) [5] as a mechanism for settling resource contention: nodes with the higher DOI receive the representations most appropriate for their state. Nodes with a low DOI receive whatever resources are left.

A DOI is calculated for each node in the network. The DOI is a function of four things: the node's a priori importance, alarm state, connectivity to other important nodes, and the perceived user interest in the node. The latter is determined by user actions to **open** (increases DOI) and **close** (decreases DOI). Funke et al. describe a similar formula for calculating the importance of a tiled window in an intelligent window management system[4].

The knowledge encapsulated within the reasoning agents is therefore:

1. how to calculate the DOI, i.e.,what constitutes importance from the standpoint of the domain and user interaction;
2. the appropriate mapping from alarm state to node representation;
3. constraints on the allocation of interface resources to node representations; and,
4. how to best suggest a representation for an opened node given the knowledge of 1-3.

Intelligent Zoom = Continuous Zoom + Intelligent Agents

The interface is affected by two sources: the SCS and the user. SCS information comes as either status (for example, alarms) or data. The user changes the interface by enacting UI controls on cluster and leaf nodes. There are five atomic user actions to control the interface:

1. **open** a node (all) – turns the node transparent, enlarges it, and displays its contents. In the case of a leaf node, the contents will be in the representation suggested by the reasoning agents;
2. **open as** a specific representation (leaf) – opens the specified representation and resizes as in (1);
3. **close** a node (all) – opacifies and shrinks the node;
4. make a node larger (**zoom in**); and
5. make a node smaller (**zoom out**).

While the system recommends representation strategies, we don't force these on the user. The system does not change the primary representations of the nodes without the user's explicit permission – hence the distinction between (1) and (2) above.

A new DOI will result if either the *system state* of the node (according to status from the control system and domain reasoning components) or its *user state* (as when the user opens or closes it) changes. In cases where opening a node involves resource mediation (such as in the command **open**, or in the example of the user requesting more than one video representation when only one may be active at a time), the interface must determine an appropriate substitute representation, and suggest it. Note that of course there are pathological cases where the requirements cannot be satisfied with respect to space, and the interface has to prompt the user to free up some space, indicating the most likely (i.e., lowest DOI) candidate nodes to be closed or reduced.

Because of the way the zoom works, any change in a node's size results in changes to other nodes' sizes as well. The interface detects when any node representation falls below a minimum size threshold and suggests an alternate representation.

There are thus two loosely-coupled tracks of concurrent behaviour in the IZ. User actions cause recalculations of the DOIs and representations subsequently suggested by the system; changing DOIs (from system status) and the indication thereof will suggest certain user actions (such as opening or closing nodes). The resulting "feel" is one of symbiotic (user and system) effort in the task of manipulating the complex data environment.

Agent Dialogues

The objective is to provide the user with the most appropriate representation for a node based on domain information, user and system resources. As stated above, the IAs evaluate both the state of domain, user and system resources, and the interface context; then a representation for each node is suggested to the graphics

agent. The zoom software constitutes the graphics agent (henceforth **GA**): it maintains the screen space and attempts to satisfy the requests for representation. These agents negotiate until an acceptable solution is reached, basing decisions on the DOI, as follows.

1. The IAs send the DOI and representation values for each node whose DOI and/or suggested representation has changed since the last communication.
2. The GA sorts all nodes on their assigned DOIs, and then attempts to render the representations suggested starting from the highest DOI. Note that its sole criterion for satisfying the request is whether the minimum space requirements of the representation can be allocated to the node: thus, the GA is the agent controlling the space resource.
3. If the GA cannot allocate enough space to the node, it requests another representation.

In this way all agents use the DOI as a basis for resource allocation.

For each (leaf) node, the IAs calculate a DOI and the (first choice) representation, and communicate this to the GA. Only the GA has any notion of hierarchy and the intermediate cluster nodes: the IAs reason only about the actual logical nodes representing monitored points in the controlled physical system. The GA keeps track of all DOIs: leaf node DOIs are explicitly sent, while intermediate nodes "inherit" the DOI of their most interesting (highest DOI) child. The GA sorts the list of all nodes on their relative DOIs and attempts to satisfy the spatial requirements of the suggested representation for each node. It does so by working "down" the list (from most to least important), first allotting space to the most important node, and then gradually reclaiming space for each successive candidate from the space allocated to the preceding more important nodes until either the current node gets sufficient space or the representation spatial needs of a more important node would be breached. If there is insufficient screen space to render the representation the GA "fails", requesting another representation suggestion for the node in question. Since there is always a representation that can fit into the minimum size of a node, and because the procedure does not backtrack to preceding nodes in the list, this implements a tractable algorithm for allocating screen space.

Architecture

Traditional architectures for intelligent user interfaces have centred around what we call the *two monoliths* approach. In this case, the interface is comprised of two sharply delineated components: an expert system (ES) and a graphical user interface[1][3][14][17]. The two are separate processes that communicate via some arbitrary protocol. The ES is often implemented using a commercial rule-based shell and populated with human factors knowledge about interface configuration; this knowledge base determines the GUI content while the system is running. The popularity of this approach likely resulted from a convergence of AI and GUI practitioners, but without a resulting unification of approach: appending an ES to a GUI seemed easier than developing a new paradigm.† Related approaches to context-sensitive GUIs have large constraint solvers embedded in the GUI code itself, making the GUI difficult to maintain and extend, and limiting the reasoning functionality to that which can produce a tolerable response time[8][10].

Such architectures result in systems where the ES – in response to external events – is continually queried about the state of the GUI. The ES is an attempt at a complete theory of the GUI; a repository of everything known about how the interface should be configured in any given situation. Constructing such a knowledge base requires an extensive knowledge engineering and knowledge validation effort.

The two monoliths approach has many shortcomings. The severest of these problems is timeliness: because the reasoning is in the form of a backtracking search procedure, a guaranteed response time is either impossible or costly in terms of solution quality. Since interactive time-critical systems (like SCS's) are primarily driven by external events (user *and* process), responsiveness is fundamental. One might therefore ask, is search appropriate at all? Perhaps the AI paradigm of "problem solving is search" is entirely inappropriate.

There are other problems with the two monoliths:

1. Redundancy of representation: both the ES and the GUI require a model of the domain and a model of the interface, which are therefore redundantly represented and updated in both components.
2. Cost: expert system shells – particularly those that claim to support real-time reasoning – can be very expensive. As well, their astonishing appetite for memory and CPU cycles can dictate a necessity for distribution to a separate workstation, thus doubling the hardware costs.
3. Delineation of task: systems built upon this approach will tend to place all reasoning within the ES, regardless of whether or not it is more appropriate to leave some of the reasoning to

† Borenstein describes this phenomenon as the result of AI researchers looking to HCI as an outlet for their view of AI without any real concern for the problem itself[2].

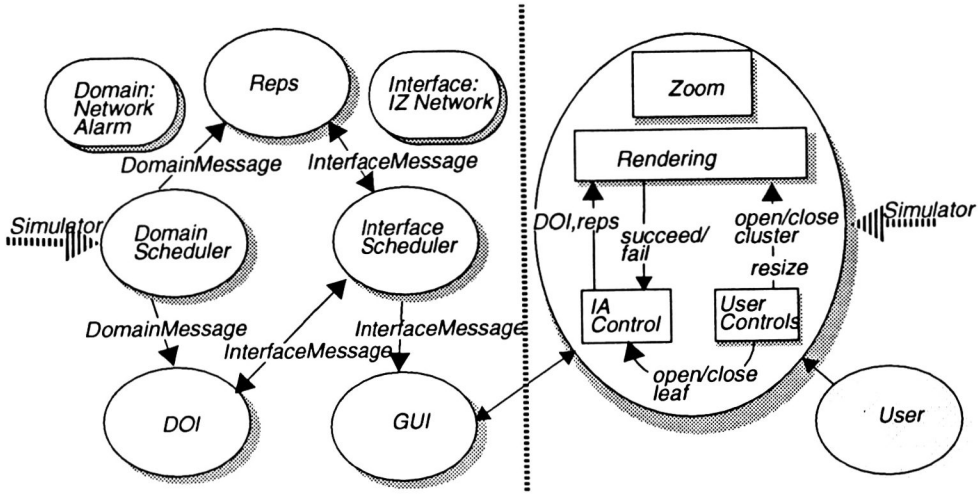


FIGURE 2. The Architecture of the Intelligent Zoom.

the GUI itself. For example, 2D layout of interface components is best reasoned about using the graphics algorithms within the GUI.

4. The ES artifact: the *event-query-response* cycle – whereby an event occurs, the ES reasons about it, and then the interface changes – reveals the ES as an obvious artifact of the interface. (Explicit menus of "ES suggestions" are an extreme example of this phenomenon.) This often results in the effect of "clumsy automation"[19], where the interface complexity is exacerbated by the need to manoeuvre around the extra functionality.

We believe that the first step towards a graceful adaptive interface is to make the expert system disappear. This requires substantial re-thinking of the two monoliths paradigm. Rather than encapsulate complete knowledge of the interface in a separate reasoning component, we argue that "intelligence" should be embedded within the graphics techniques and distributed as independent processes that manipulate the objects comprising the interface.

In this approach, "intelligence" takes the form of a collection of simple, autonomous knowledge sources (also called agents, or tasks or processes because of their natural correspondence to multiprocessing entities). Effective behaviour results from the sum of the parts, not because of one single part. This type of design falls into the category of AI *blackboard systems* [7] and is consistent with modern approaches to real-time operating systems[18]. There are three components to blackboard architectures: blackboards, knowledge sources, and schedulers. A *blackboard* is a globally accessible data structure. Blackboards are read from and written to by knowledge sources. A *knowledge source* is an independent task that encapsulates some small piece of procedural or declarative intelligence. The knowledge sources only communicate with each other through changes to the blackboard(s). A *scheduler* is responsible for serializing the inherently parallel execution of the system. Schedulers invoke knowledge sources, act as semaphores for the blackboards, and ensure real-time response.A task is only scheduled if a technique it supports is presently active and if the message content will be of interest. The scheduler must also decide the order in which the tasks are executed, and, in the face of increased system activity, whether to execute certain (slower) tasks at all

The knowledge sources are independent tasks that read from and write to this collection of blackboards. The key concept is that each GUI technique is encapsulated as a set of tasks that together render the technique, capture user interaction, and reason about its inter-contextual presentation.

The advantage of the blackboard approach is the resulting unified view of graphics and intelligence simply as tasks interacting with common data structures. A simplified software development environment results. The strict delineation of tasks and the ES artifact are gone: knowledge is free to reside where appropriate. Efficient performance is achieved as the schedulers can gracefully degrade the activation of less critical tasks during high tempo operation. Moreover, it is relatively easy to reconfigure the behaviour of the interface as it simply requires a change to a task, the removal of a task, or the addition of a task. Since the tasks are isolated and independent, the problem of interfering and inconsistent knowledge endemic to large rule-based expert systems

can be, if not defeated, at least axiomatised.

Implementation

The implementation of the intelligent zoom incorporates the continuous zoom in the graphics agent and an intelligent mediation (IM) approach using the blackboard architecture described above. In the first development phase the tools we needed were unavailable on a single platform: graphics development was done on an SGI, while the reasoning components were realised on an HP720, where the best multitasking support resided.

Figure 2 shows the current architecture, where ovals represent **tasks**, rounded boxes are **blackboards** and rectangles procedural modules.

The graphics agent of the IZ is a single Unix process with two concurrent "threads". User actions directly manipulate the zoom and affect the state of the interface; the IA events set the DOI and representations which are then used to reconfigure the parts of the interface not directly controlled by the user.

When Rendering is invoked, it asks the Zoom Algorithm to redistribute the space such that the most important nodes get "first crack" at their desired representations. If it cannot, the GA **fails** and sends a message back to the GUI Handler, requesting another representation suggestion for the node in question. Since display changes are usually close to incremental (changes are not requested for every leaf node at each message or user request), the GA need only sort the DOI table, compare current representation assignments with the requested, and pick up in the DOI list at the point where these diverge, since it does not backtrack to the preceding node allocations. This contributes to efficient performance.

Together with the GA, the Reps and DOI tasks embody the "intelligence" in the IZ. The Reps task is responsible for mediating the representations allocated to the leaf nodes in the IZ. This task essentially embodies the mediation of representations, for it views representations as a limited interface resources. The Reps task is scheduled for execution whenever the user opens or closes a leaf node; that is, whenever representation resources are demanded or released.

During initialization of this task, a mediator for each representation is created with an indication of how many allocations are permitted, e.g., one for video, four for schematic, etc. During program execution, each mediator maintains a list of all nodes presently consuming the representation (maximum length equal to the number of permitted allocations), and a list of all nodes wishing to consume the representation. Both lists are sorted by DOI. A new request for a representation is granted if the resource is still available (i.e., not all instances are allocated) or if the requesting node has a greater DOI than any currently using the representation. In the latter case, the representation is "taken away" from the lower-DOI node and the negotiation phase restarted for its alternate representation.

The Reps task also contains knowledge in the form of a mapping from node state to a sorted list of the most suitable representations. This list is consulted during the negotiation process when a node is first opened, or when a node must be assigned a new representation during a reallocation phase. This declarative knowledge ideally belongs in a user model given that it reflects an operator preference for certain representations under certain alarm conditions.

The IZ is therefore reasoning about representation allocation with respect to interface resources (hardware and screen space) and a limited set of user resources (where we restrict the number of representations currently visible based on guidelines of screen density).

The DOI task is responsible for calculating a DOI for each leaf node in the IZ. The DOI formula is encapsulated within this task. Because a node's DOI depends on its neighbours' DOI, changes must be propagated through the network, which is done using an arc consistency algorithm [9]. Because a node's DOI is dependent on its alarm state and user interest (i.e., has the user just opened or closed it), this task is scheduled for execution whenever a message from the GUI or the Simulator is received by the respective schedulers.

Discussion

The first version of the intelligent zoom is prototypical in nature and as such does not contain the extent of knowledge one would expect in a real system. Nonetheless, it serves to illustrate how effective active contextual assistance can be in a complex, information-rich graphical interface. The marriage of maintaining overall context in the continuous zoom display while assisting in the preparation of local detailed context using adaptive presentation illustrates how a symbiosis of operator and system involvement can reduce the effort required to use a large information space. Specifically, the intelligent zoom addresses the research problems initially discussed:

- It provides a new way of visualizing and navigating large information spaces, especially well suited for hierarchical information organisation and concurrent, disparate operator and system activity.
- The IZ is a framework for the development of alternative representations for data in the context of multiple displays and concurrent, real-time interrupting tasks, and for the development of knowledge in

guiding the dynamic choice of best display technique given current operator, system, and interface state. It succeeds in context sensitive presentation because the interface is able to reason about its own context as well as about that of the SCS and user preferences.

- The IZ is a framework for mediating dialogues between the SCS and operator (likewise between the user interface and operator) as it mediates the allocation of resources to concurrently active focus points.

The architecture is not the completely unified approach advocated above, and as such it suffers from the problem of redundant data structures and context across two machines. However, it avoids the worst pitfalls of the *two monoliths* approach. Response is quick and the interaction is graceful: there is no sense of waiting until the system "makes up its mind".We have run the IZ on a variety of network configurations, both locally (both machines at SFU) and remotely (one machine at SFU, and one in Calgary). Practice has shown performance to be well within user tolerance (within several frame times running on two local machines, and within a half-second across provincial boundaries).

Studies are underway to examine the use of active context-based assistance in the IZ under varying workload control room conditions. Planned and in the design stage are studies to test the IZ against other network navigation techniques and to determine the level of adaptivity operators prefer.

Acknowledgements

This research is part of the Intelligent Graphic Interface (IGI) project sponsored by the PRECARN Associates consortium of Canadian industrial and research organizations, and we are indebted to the contributions of our colleagues on the IGI Industrial Prototype Team at MPR TelTech Ltd. We gratefully acknowledge the generous support of the govenments of Canada, the Province of British Columbia, and of PRECARN Associates.

References

[1] Yigal Arens, Lawrence Miller, and Norman Sondheimer. "Presentation Design Using an Integrated Knowledge Base." In Joseph W. Sullivan and Sherman W. Tyler, eds. *Intelligent User Interfaces.* New York: ACM Press, 1991, pp. 241-258.

[2] Nathaniel S. Borenstein. *Programming As If People Mattered: Friendly Programs, Software Engineering, and Other Noble Delusions.* Princeton, N.J.: Princeton University Press, 1991.

[3] M. H. Chignell and P. A. Hancock. "Intelligent Interface Design." In M. Helander, ed. *Handbook of Human-Computer Interaction.* Elsevier Science Publishers, 1988, pp. 969-995.

[4] Douglas J. Funke, Jeannette G. Neal, and Rajendra D. Paul. "An Approach to Intelligent Automated Window Management." *Int. Journal of Man-Machine Studies* (1993), **38**, pp. 949-983.

[5] G.W. Furnas. "Generalized Fisheye Views". *Proceedings of the ACM CHI '86 Conference on Human Factors in Computing Systems.* pp. 16-23.

[6] W.E. Gilmore, D.I. Gertman, and H.S. Blackman. *User-Computer Interfaces in Process Control: A Human Factors Engineering Handbook.* Academic Press, San Diego, 1989.

[7] V. Jagannathan, Rajendra Dodhiawala, and Lawrence S. Baum, eds. *Blackboard Architectures and Applications.* Boston: Academic Press, Inc., 1989.

[8] S. Kochhar. "A Prototype System for Design Automation via the Browsing Paradigm". *Proceedings of Graphics Interface 1990*, pp. 156-166.

[9] Alan K. Mackworth. "Consistency in Networks of Relations." *Artificial Intelligence,* 8, pp. 99-118.

[10] David L. Maulsby, Ian H. Witten and Kenneth A.Kittlitz. "MetaMouse: Specifying Graphical Procedures by Example." *Computer Graphics,* **23**(3), pp. 127-137.

[11] C.M. Mitchell. "GT-MSOCC: A Domain for Research on Human-Computer Interaction and Decision Aiding in Supervisory Control Systems." *IEEE Transactions on Systems, Man, and Cybernetics*, 17(4), pp. 553-572.

[12] E.G Noik. "Layout-independent Fisheye Views of Nested Graphs". *Proceedings of the 1993 IEEE Symposium on Visual Languages*, Bergen, 1993, pp. 336-341.

[13] Russell Ovans and William S. Havens. "Intelligent Mediation: An Architecture for the Real-Time Allocation of Interface Resources." *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, Orlando, 1993, pp. 55-61.

[14] William B. Rouse, Norman D. Geddes, and Renwick E. Curry. "An Architecture for Intelligent Interfaces: Outline of an Approach to Supporting Operators of Complex Systems." *Human-Computer Interaction*, 1987-88, **3**, pp. 87-122.

[15] Manojit Sarkar and Marc H. Brown. "Graphical Fisheye Views of Graphs". *Proceedings of ACM CHI '92 Conference on Human Factors in Computing Sys-*

tems, pp. 83-92.

[16] Doug Schaffer, Zhengping Zuo, Lyn Bartram, John Dill, Shelli Dubs, Saul Greenberg and Mark Roseman. "Comparing fisheye and full-zoom techniques for navigation of hierarchically clustered networks". *Proceedings of Graphics Interface 1993*, pp. 87-97.

[17] Doree Duncan Seligmann and Steven Feiner. "Automated Generation of Intent-Based 3D Illustrations." *Computer Graphics*, **25** (4), pp. 123-132.

[18] Andrew S. Tanenbaum. Modern Operating Systems, Prentice-Hall, N.J., 1992.

[19] David Woods. "The Price of Flexibility." *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, Orlando, 1993, pp. 19-25.