

Task-Sensitive User Interfaces: grounding information provision within the context of the user's activity

Nathalie Colineau, Andrew Lampert and Cécile Paris

CSIRO – ICT Centre

Locked Bag 17, North Ryde NSW 1670, Australia

phone: (+61) 2 9325 3100

[\[nathalie.colineau, andrew.lampert, cecile.paris}@csiro.au](mailto:{nathalie.colineau, andrew.lampert, cecile.paris}@csiro.au)

ABSTRACT

In the context of innovative Airborne Early Warning and Control (AEW&C) platform capabilities, we are building an environment that can support the generation of information tailored to operators' tasks. The challenging issues here are to improve the methods for managing information delivery to the operators, and thus provide them with high-value information on their display whilst avoiding noise and clutter. To this end, we enhance the operator's graphical interface with information delivery mechanisms that support maintenance of situation awareness and improving efficiency. We do this by proactively delivering task-relevant information.

Categories and Subject Descriptors

H.5.2 [User Interfaces]: Natural Language, Graphical User Interfaces; I.2.7 [Natural Language Processing]: Discourse, Language Generation.

General Terms

Design, Human Factors.

Keywords

Information delivery, task-sensitive interface, discourse planning, Information visualization.

1. INTRODUCTION

In the context of innovative Airborne Early Warning and Control (AEW&C) platforms, we investigate a task-sensitive user interface that provides air combat operators with coherent and tailored displays. In AEW&C systems in general, operators are interacting in a complex environment, one characterized by large volumes of information and the need to integrate information from a variety of sources. This can lead to visual clutter on the display and a high cognitive load for operators, who must memorize information in order to understand the situation. To overcome the operators' information overload and support them in their tasks, our approach consists of enhancing the operators'

graphical interface with information delivery mechanisms that tailor this complex environment to their tasks and information needs. To do so, the system observes the operator's activity and delivers appropriate information through its delivery mechanisms, ensuring that the most relevant information is always available and prominently displayed. Thus, this approach presents the advantage of grounding the information provision within the task-at-hand.

Similar approaches have been developed in various domains: the attentive information system suggests to users potentially useful information close to their current interest [9]; the Watson information access system observes users interacting with applications and anticipates their information needs using a task model [2]; the CodeBroker system observes programmers' activities and proactively delivers components that match the inferred needs [15]; the Pilot Vehicle Interface automatically selects display formats and control functions tailored to the pilots' tasks and situations ([7], [14]). In each case, based on the observation of the user's activities, these systems provide contextualized information relevant to tasks and users. However, in contrast to those systems, our approach does not only provide relevant information, but also presents the information coherently through appropriate media and integrates it smoothly with the elements of information already on the screen. Based on linguistic theories about discourse, relevance and coherence, the system selects, organizes and designs the presentation of multimedia information to be integrated on the display. Thus, the information delivery mechanisms we propose, when integrated with the operators' interface, enable the latter to become sensitive to the operator's task environment.

In this article, we show how a graphical user interface (GUI) can be integrated with delivery mechanisms to allow task-sensitive information delivery. In particular, we present our delivery engine, the Virtual Document Planner (VDP), which is responsible for automatically generating multimedia information for the operators. We first introduce the air combat operators' environment. Then, we discuss the concept of a task-sensitive user interface, and use a specific operator's task as a running example to illustrate how to generate task-tailored information and how to enable the GUI to adapt the display to support the operator in the task-at-hand.

2. THE SURVEILLANCE OPERATOR'S ENVIRONMENT

The air defense organization relies on information about the battle space to complete their job. The primary source of information is real-time information from sensors such as radars that indicate the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Advanced Visual Interfaces (AVI 2004) May 25–28, 2004, Gallipoli, Italy.

Copyright 2004 ACM 1-58113-000-0/00/0004...\$5.00.

location of airborne objects in the area. This radar information can be difficult to interpret in its raw form. A tracker system can, over time, associate radar returns as coming from an individual aircraft. This aircraft can then be established as a track. The raw radar picture and track symbols are the basic display seen by air defense operators. In addition to the sensor's return, other information may come from a variety of data sources. It is the role of the surveillance operator to collect and integrate this additional information to facilitate the interpretation of the battle space. The combined output of the surveillance operator's work is the *Recognized Air Picture*. It is considered as the best understanding of the current situation in the battle space and is used by fighter controllers and command and control (C2) personnel to complete their own tasks.

As mentioned above, the surveillance operator's role is to convert the radar returns and basic track information into an information-rich picture of the battle space. When a surveillance mission begins, there are a large number of tracks (typically in the order of 200-300 tracks) that appear on the radar screen. There will be little or no information associated with the tracks, and, if passed along at this stage, that information would be of limited use for fighter controllers and C2 personnel. The surveillance operators' primary activities are to classify tracks based on their threat assessment (e.g., Friendly, Hostile, Neutral or Unknown), and to identify them. The aim of the identification task is to provide enough information about individual tracks to make them distinct from each other, and to allow fighter controllers (i.e., the operators for whom the *Recognized Air Picture* is built) to make decisions about whether and how the track is to be intercepted. It is not expected that a final and complete air picture will be achieved; rather it will require constant maintenance to ensure that track information continues to be accurate.

Surveillance operators work with information that comes from various sources and that may be visualized on different displays. For example, they have access through an extra terminal to a database of international flight plans available on the Airborne Fixed Telecommunications Network (AFTN). Details from the AFTN are viewed and then manually entered into the system. Considering this environment, to fulfill the operators' needs, it is important to first provide them with an easy way to access and integrate information in one display, and also to select and display only information tailored to the task-at-hand.

3. TASK-SENSITIVE DELIVERY

As briefly described in section 2, air combat operators interact with a highly variable information environment, where it is important not to deliver all information that is available. Indeed, this could result both in information overload and in the display of information that is not currently needed by the operator. In contrast, operators need a display that answers their information needs, providing information tailored to their tasks and allowing them to focus on important information.

To this end, we have developed an approach based on a representation of the operators' task and the context in which they perform their task. This drives the interaction process and allows the interface to provide operators with relevant information. To implement this approach, we combined our information delivery engine, the Virtual Document Planner (VDP), with the graphical user interface (GUI). By integrating the VDP with the GUI, we

ground the delivery of information within the context of the activities the operator is performing. The GUI becomes task sensitive by proactively and appropriately delivering the information required at each step of the operator's task.

3.1 Virtual Document Planner

The Virtual Document Planner (VDP) is our delivery engine [3]. It generates multimedia information tailored to operators' tasks. The VDP is based on a typical Natural Language Generation (NLG) architecture, where the linguistic resources are separate from the engine. The engine uses discourse rules, which select and organize the content, design rules, which determine an appropriate way to realize the content and the structure of the presentation, and a set of delivery functions to realize the media objects to be presented.

The specific engine that the VDP employs is a planning engine, and the resources are represented as plans, as in [12]. The VDP employs a discourse generation approach. In this approach, a communicative goal (or discourse goal) is given as input to the planning engine. The communicative goal is to answer an information need of the user, the operator in this case. The planning engine then tries to achieve this goal, using its set of discourse and design rules, using a traditional goal/sub-goal decomposition. In addition, this approach exploits rhetorical relations (or coherence relations), based on Rhetorical Structure Theory (RST) [10], to guarantee the coherence of the resulting presentation. Coherence relations must hold between sibling sub-goals in the goal decomposition. They indicate how the various discourse segments and piece of information fit together to achieve the communicative goal.

The process of designing and delivering information has to go through several decision steps to decide what needs to be conveyed (i.e., what to say and what to show) and the most appropriate way to convey it (i.e., how to present the information). The generation process proceeds in four main steps:

- The design of the discourse structure determines what needs to be communicated by selecting and organizing relevant information from a variety of sources;
- The design of the presentation structure determines how the information has to be presented, namely how each discourse segment should be realized through specific modality (e.g., text, graphics);
- The media object realization builds each media object and links them together to form a coherent presentation. This is then passed to the GUI, which interprets it;
- The contextualization of information consists of integrating the content to be presented within the current situation, namely what is already displayed on screen.

3.2 Graphical User Interface (GUI)

The GUI is the interface through which operators interact with the system. It is a prototype, which implements a subset of features that allow us to demonstrate the technology without seeking absolute real-world verisimilitude. Our GUI is implemented in Java. The display of spatial information is handled using the OpenMap toolkit (<http://openmap.bbn.com>).

The primary input to the GUI is from a simulator module, which generates tracks (i.e., simulated series of radar returns, which are associated with a single individual aircraft). Tracks are displayed on the GUI using standard military symbology (MIL-STD-2525B). In addition, the GUI uses a series of data stores to maintain information about objects within a simulation (e.g., tracks, flight plans). These data stores contain both type and instance information about these objects and provide the domain model for our simulated world. This domain knowledge is one information source used by the VDP during planning.

Although the GUI and the VDP share some knowledge resources, they are loosely coupled components that have distinct roles: the VDP designs the information to be presented and the GUI renders it on the operators' displays. Thus, we need to define a flexible Application Programming Interface (API) for the GUI, which enables the VDP to deliver not just content, but also to specify visualization and interaction constraints. To this end, we use BeanShell (<http://www.beanshell.org>), a free Java source interpreter. By using the BeanShell interpreter, we create a very flexible scripting interface for the GUI. Indeed, with this API, the VDP can decide when and how selected content is displayed on the GUI. This control is exercised through the delivery of scripted commands (using a superset of Java syntax) to the GUI. These commands are dynamically interpreted by the BeanShell interpreter. Moreover, as the interpreter runs in the same Java Virtual Machine (JVM) as our GUI, it can access and manipulate any existing run-time instance of any class, or even define new classes dynamically. For instance, it becomes possible to modify existing object instances, instantiate new instances of existing classes or define new classes on the fly. Thus, this allows us to have a very generic GUI and to dynamically define combinations of primitive Swing and OpenMap features to provide flexible visualization techniques. New visualization techniques can also be added with minimal or no code changes to the GUI.

By observing the operator interacting with the GUI, the system

can infer the task being undertaken. This provides the VDP with a context and a reason to select, organize, and deliver appropriate content to the GUI. We exploit these capabilities to create an intelligent, task sensitive display. As the operator interacts with the GUI, the display is adapted by the VDP, which issues commands to the GUI API, providing task-specific information. To guarantee the consistency of the display, the GUI's current state is taken into account at runtime when information is displayed. This allows the display to be gradually transformed to provide new information. Furthermore, this ensures that the system provides smooth transitions between display configurations for the operator.

4. THE DESIGN OF INFORMATION PROVISION

This section details the main steps that drive the generation of task-tailored information through a running example. This example illustrates the information the system would proactively provide to the operators in an identification task (cf. section 2).

4.1 The discourse structure

To plan information provision, we need to start with a communicative goal, that is we need to know the reason why information needs to be conveyed. In this case, as information is provided to support operators in their tasks, the communicative goal of particular information may be to enable operators to complete a task, to allow them to interpret a situation or to provide them with additional details on a particular track. Once the communicative goal is known (the role of the information to be conveyed is established), the process of selecting and organizing the appropriate information content can start.

Let's say for this example that a track has been classified neutral by the operator. Based on the operators' task analysis, the system knows that there is a need now to provide identification features

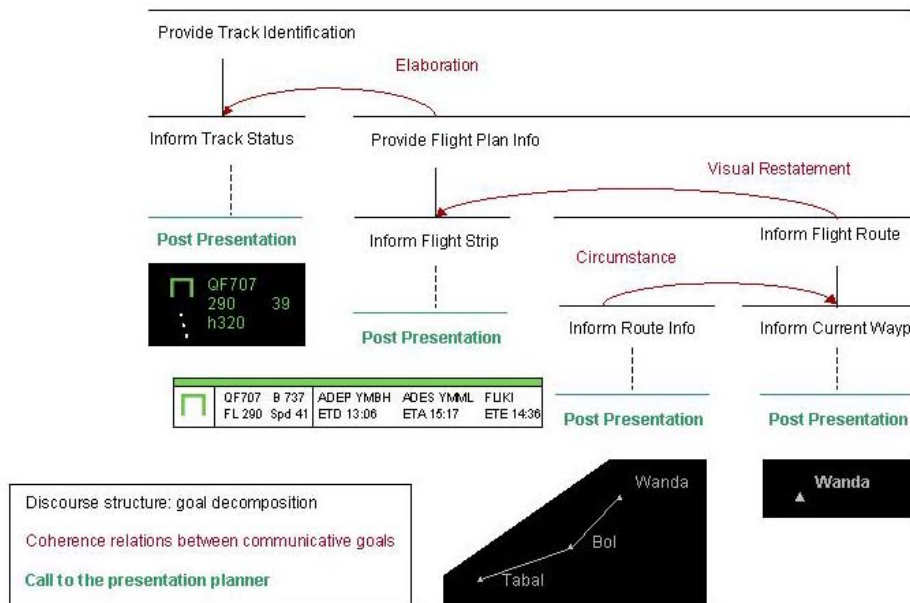


Figure 1. Discourse structure for the track identification

Table 1. Example of discourse plan operator

```
<operator>
  <id>FlightRouteInformation</id>
  <description>Display flight route information</description>
  <effect>(Inform FlightRoute ?flight ?flightplan)</effect>
  <constraint>(set ?wpt (retrieval:getCurrentWpt ?flightplan))</constraint>
  <nucleus>
    <value>(Inform CurrentWaypoint ?wpt ?flight)</value>
  </nucleus>
  <satellite>
    <relation>circumstance</relation>
    <value>(Inform RouteInfo ?flightplan)</value>
  </satellite>
</operator>
```

for this track. This will constitute the communicative goal and the VDP will have to plan information to satisfy this goal: provide track identification. When receiving a communicative goal to satisfy, the VDP has first to match it with a discourse strategy. The discourse strategies decompose a complex communicative goal such as providing identification features for a specific track into sub-goals, until there is no further possible decomposition (i.e., until reaching elementary discourse goals such as Inform ?x). The resulting representation is a hierarchical discourse structure that represents and organizes the content to be conveyed. In particular, this discourse structure explicitly represents the coherence relationships between discourse segments (or content elements). Figure 1 shows the discourse structure that corresponds to the communicative goal: provide track identification. Note that the graphical representations at the bottom of the tree are not part of the discourse structure. They are here to give an illustration of what will be realized by each discourse segment. The content conveyed by the discourse structure is organized around two parts, one constituting an elaboration of the other one:

- A simple discourse segment, Inform track status, which consists of visualizing on the map the track symbol and its kinematics information; and
- A complex discourse segment, Provide flight plan information, which is further decomposed. The flight plan information provided is split into two parts: the flight plan itself, displayed in a strip window, and the flight route, visualized on the graphical display. The coherence relations make explicit the role of each piece of information with respect to the whole. In this case, the flight route acts as a visual-restatement (i.e., as a graphical support) of the flight plan. Finally, the flight route itself is further decomposed. The current waypoint, namely the next beacon the aircraft is about to reach, is visualized and emphasized on the graphical display. The complete aircraft route is provided as an interpretative context for the current waypoint.

The discourse structure represented in Figure 1 constitutes the output of the content planner. This structure is built using discourse strategies (also called discourse plan operators in a discourse planning approach). They describe how to achieve a communicative goal, specifying how the discourse should be organized and what type of information should be included. Thus, the role of the information to be presented is central in this approach as it drives the discourse structure planning and the reasoning on how the media objects will be displayed and related together on the GUI. Although the discourse structure appropriate to support a particular task is encoded in the discourse strategies, the discourse structure is not completely pre-defined in advance,

but built at runtime. Indeed, several discourse strategies may be available for the same communicative goal. For example, one discourse strategy may provide the minimal information while another one may provide all the content available on a particular topic. In our context, content is selected depending on the specific classification assigned to the track. So, based on the type of track, the system will select and deliver different identification features. As is typical of generation and multimedia systems to-date (e.g., [1]; [5]; [11]), the discourse strategies implemented also allow the system to control what and how much information is to be provided. They have been designed from the analysis of the operators' tasks and information needs, which have been formalized into a task model (cf. [4]).

An example of a discourse strategy is provided in Table 1. A discourse strategy is represented by a name, a description, an effect, some constraints, a main sub-goal (called nucleus in RST terms), and one or more subsidiary sub-goals (called satellite in RST terms). The effect of a discourse strategy is the communicative goal it can satisfy. If the strategy succeeds, the communicative goal is satisfied. The constraints define the applicability conditions of the strategy, namely when it may be used. Constraints can be on a number of criteria: the topic under discussion, the data available, the user model, the discourse history, etc. The discourse strategy in Table 1 develops the discourse segment related to the flight route display. The goal to satisfy is: Inform FlightRoute ?flight ?flightplan, the main sub-goal is: Inform CurrentWaypoint ?wpt ?flight, complemented by a subsidiary sub-goal, Inform RouteInfo ?flightplan. This will provide circumstantial information to the nucleus, that is a framework for interpreting the nucleus.

To decide which discourse strategy to apply, the system checks the conditions of applicability of each strategy that are expressed in the constraints. Discourse strategies are applied in turn until each sub-goal is decomposed. During this process, content is selected and organized. The resulting representation is a tree structure where each leaf represents a discourse segment, namely a piece of information to be presented.

4.2 The Presentation structure

Once the discourse structure has been built, the VDP needs to specify how each discourse segment should be realized. The VDP thus has to match each discourse segment to be presented with a presentation strategy. The presentation strategies plan a presentation for each discourse segment, specifying which part should be realized by one modality or another (i.e., text or graphic). The resulting representation is a hierarchical

presentation structure that represents how the content will be conveyed (i.e., in which modality it has to be encoded), as illustrated in Figure 2. This figure shows only the presentation structure of a waypoint, as the complete design corresponding to the discourse structure shown in Figure 1 would have been too complex to represent in one figure. In the presentation structure (cf. Figure 2), the arcs are labeled with design relations. The leaf nodes represent the sub-goals sent to the realization module.

The presentation structure shown in Figure 2 constitutes an example of how the content might be designed, and the picture at the bottom right (same figure) shows the resulting display. This structure also highlights the way the media objects will be combined within the presentation (e.g., association by collocation, connection, attachment, etc.). The presentation structure corresponding to the waypoint design is decomposed in two parts:

- The right part corresponds to the design of the waypoint itself;
- The left part corresponds to the user interaction allowed on that particular media object and the presentation of the information triggered. Indeed, explicitly reasoning about the content to be presented and its presentation structure offers the advantage of being able to design the interaction process at the same time.

Looking back to the right side of the presentation structure in Figure 2, we can see that a waypoint is represented as a position on the map with a label. Just like the discourse relations specify how discourse parts are related to each other and thus, together, form a coherent whole, the design relations specify how elementary objects are combined into media objects and thus, together, form a coherent presentation. Table 2 presents the design relations used in the example (N means nucleus, and S means satellite). This work is based on the concept of visual task defined by Zhou and Feiner (cf. [16]). The design relation *Association:collocation* links the position to the label and specifies the way they should be visualized, namely by collocating the label with the position.

The left side of the presentation structure in Figure 2 represents additional information that has been attached to the waypoint media object. The *PopupElaboration:mouseover* design relation uses the metaphor of the hyperlink. When the operator moves the mouse over the media object, he/she triggers the presentation of additional information. In this case, the additional information is planned with the waypoint it is attached to, but its presentation is triggered by an event specified in the design relation. The additional information is displayed using the excentric labeling technique designed to label neighborhood of objects located around the cursor [6]. In addition to the provision of additional information, the waypoint is emphasized as well.

An important thing to remember is that there is typically more than one valid way to present the data. For example, a waypoint could have been represented by only a dot. Similarly, the additional information could have been visible all the time. The design of each presentation depends on a combination of several criteria, including the data available and their characteristics, the purpose for which the data are presented, and the modalities combined. Those criteria may be translated into constraints in the presentation plans to decide which design strategies are the most appropriate. For example, Rutledge *et al.*, [13], have investigated a way to translate rhetorical structure into constraints used to derive the presentation structure. Thus, they show how varied presentations can be generated from the same discourse structure.

A final point worth raising is the advantage of organizing an entire presentation as one discourse and presentation structure. This allows the system to maintain an overall coherence, even if subparts correspond to elements to be realized in different modalities. This is indeed important as both the discourse and the design relations will be used in the processes that follow the discourse and presentation planning stages. While the design relations specify how to integrate the media objects together, the discourse relations indicate their respective role in the whole presentation, enabling the GUI to create the appropriate links. Once the presentation is planned, each presentation segment is sent to the realization module in order to build the media objects.

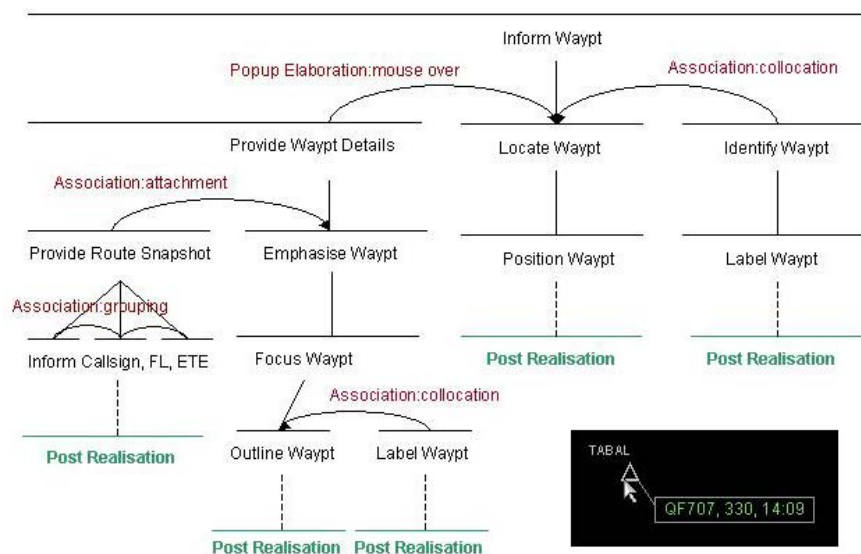
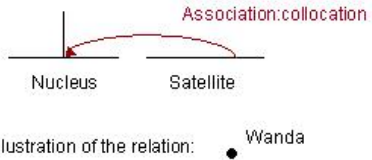
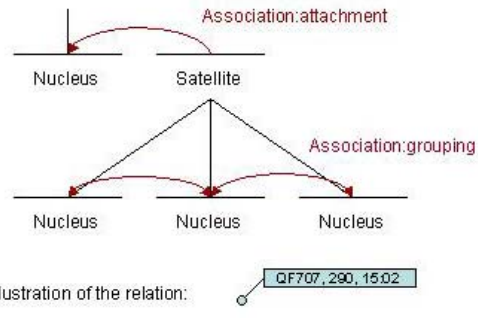
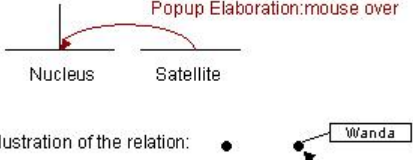


Figure 2. Presentation structure for the design of a waypoint

Table 2. Design Relations between Communicative Goals

	<p>Constraints: N is a media object with a position S is a media object without a position</p> <p>Effect: S's position will be derived from N's position. S will be collocated to N.</p>
	<p>Constraints: N is a media object with a position S is a media object without a position</p> <p>Effect: S's position will be derived from N's position. S will be attached to N as an excentric label.</p> <p>Constraints: none</p> <p>Effect: this is a multi-nucleus relation. The nuclei are presented together and grouped as a block of information.</p> <p>Note: the way the grouping is presented is usually specified elsewhere. In the opposite example, the grouping elements are then associated to a position by an attachment relation.</p>
	<p>Constraints: none</p> <p>Effect: S presents additional information about N. The presentation of S is triggered by an event. This event is here a mouse over.</p>

4.3 The Media-Object Realization

The completed presentation tree structure contains presentation segments that must be realized to produce media objects for display on the GUI. The realization of these presentation segments is performed using a series of delivery functions.

Delivery functions are parameterized functions that produce commands defining how to realize specific presentation segments, in syntax understood by the GUI API. For example, in Figure 2, *Position Waypt* is a presentation segment ready to be realized. This presentation segment calls a delivery function, (in this case the function *delivery:position*), that produces a command specifying how to render the waypoint position, as shown in Table 3. This command is then interpreted by the GUI API and the corresponding objects displayed on the GUI. The command remains high-level and independent from the specific programming language used by the GUI environment.

Table 3. Command for the rendering of the waypoint position

```
Position("YMML", "-37.67", "144.84", "map");
```

Once the delivery functions have been executed, the resulting commands produced (which encode the presentation segment realization) are annotated back into the presentation tree structure. At this stage, only the elementary media objects have been constructed. The system now needs to assemble elementary media objects together to form a complete and coherent presentation. For example, the flight route is a media object composed of several elementary media objects including waypoints and the departure and destination airports. In turn, a waypoint is composed of elementary media objects including the waypoint position and its textual label. Thus, the presentation of the flight route media

object results from the coherent integration of all these media objects.

To decide on how to assemble media objects together, we use the information encoded in the relations of the tree structure. Indeed, as the role of each piece of information and its layout is specified by relations in the tree structure (through both discourse and design relations), a final top-down pass through the tree structure has to be done to determine further context for realizing the content. This is performed by our realization module. It examines both the discourse and the design relations, as its objective is two-fold. The first objective is to reason about the design relations to determine exactly how the realization output from the delivery functions should be combined before being passed to the GUI. For example, if we consider the waypoint presentation structure (shown in Figure 2), the waypoint in itself (right part of the structure) is composed of two elementary media objects (the position and the label) that are realized by two separate delivery functions. The final top-down pass aims at interpreting the *Association:collocation* design relation to link those two elementary media objects together and build a waypoint object. When the design relation is parsed and interpreted, the label is assigned a position collocated with the waypoint position as shown in Table 4.

Table 4. Command for the rendering of a waypoint with label

```
Position("YMML", "-37.67", "144.84", "map");
sat = FormattedLabel("Melbourne Airport");
sat.setPosition("-37.67", "144.84");
```

The second objective is to reason about the discourse relations. This consists of linking the media objects constituting the presentation according to the role they have in the discourse

structure. Indeed, if we consider the discourse structure in Figure 1, there are four media objects: the track, the flight strip, the flight route and the current waypoint. Each of these has a specific role in the discourse structure, which is useful to define how these objects should be linked together and behave in a consistent way. For example, based on the fact that the flight plan is providing an *elaboration* on the track status, if the operator clicks on the track, this will automatically trigger the display of the flight plan information. Similarly, as the flight route is a *visual-restatement* of the flight strip, when the operator maximizes the flight strip, the flight route is visualized on the map, and when the operator minimizes the flight strip, the flight route is hidden. Thus, instead of linking media objects arbitrarily and hard coding those links in the GUI, the relations are dynamically built by interpreting the discourse relations embedded in the tree structure. Similar linking has been implemented in the Electronic Program Guide (EPG) system [13], an application for movies broadcast on television. Each movie is linked with a text of description using the *summary* rhetorical relation. When the user clicks on a movie image, this triggers the associated summary based on the rhetorical relation that links those two objects.

Once the realization module has completed its reasoning, a complete set of commands, which we term a presentation, is ready to be passed to the GUI. This presentation is generally contained in a single file, though could be spread across multiple files. This file is processed by the GUI interpreter, and the defined actions are realized on the graphical display. Figure 3 illustrates the display resulting of the discourse and the partial presentation structure shown in Figure 1 and Figure 2.

4.4 The Delivery within the Situation

The final step in this delivery process is to contextualize the presentation, namely smoothly integrating the information to be presented with what is already on screen. This raises some issues and we report here our initial investigation. When providing new information on the display, it is important that the information

provided is not just added to but really integrated within the display. This presents several advantages. First, the coherence and the relevance of the information presented is maintained and related to the situation. Indeed, as already mentioned, the role of the information to be presented is central in this approach as it drives the delivery process and the interaction process. Moreover, smooth transitions are provided between presentations. When dealing with interactions, coherence issues do not only occur within a single presentation but have to be considered throughout the interaction process, namely across displays. Finally, the display looks less cluttered and more consistent from task to task. Thus, it is important to know what is already on screen (i.e., how crowded is the display) to make an appropriate decision. This enables the system to decide, for example, what information needs to be hidden, de-emphasized or minimized, to present prominently what is now relevant to the current task.

Our approach to delivering information is not to impose information on operators but to send them a notification in the workspace located near their graphical situation display. The notification is represented by a minimized window containing the information relevant to the task-at-hand. In this way, operators know that new information is available and can consult it when appropriate for them. Indeed, it is important to ensure that the information provided does not interrupt operators in their task. By proactively delivering information that has not been explicitly requested by the operator, the system potentially risks distracting operators' focus of attention. To overcome the potential intrusiveness of unsolicited delivery, it has been shown [8] that the system should balance the cost of interrupting users in their task with the loss of context-sensitive delivery by deferring the provision of information. Thus, to keep the advantage of task-sensitive delivery, our system needs to work out a way to gently notify the operators that new information is available, without necessarily interrupting them in their task. The approach envisaged is similar to what has been developed in the CodeBroker system [15]. First, the information provided does not require any response from the operators. The information is

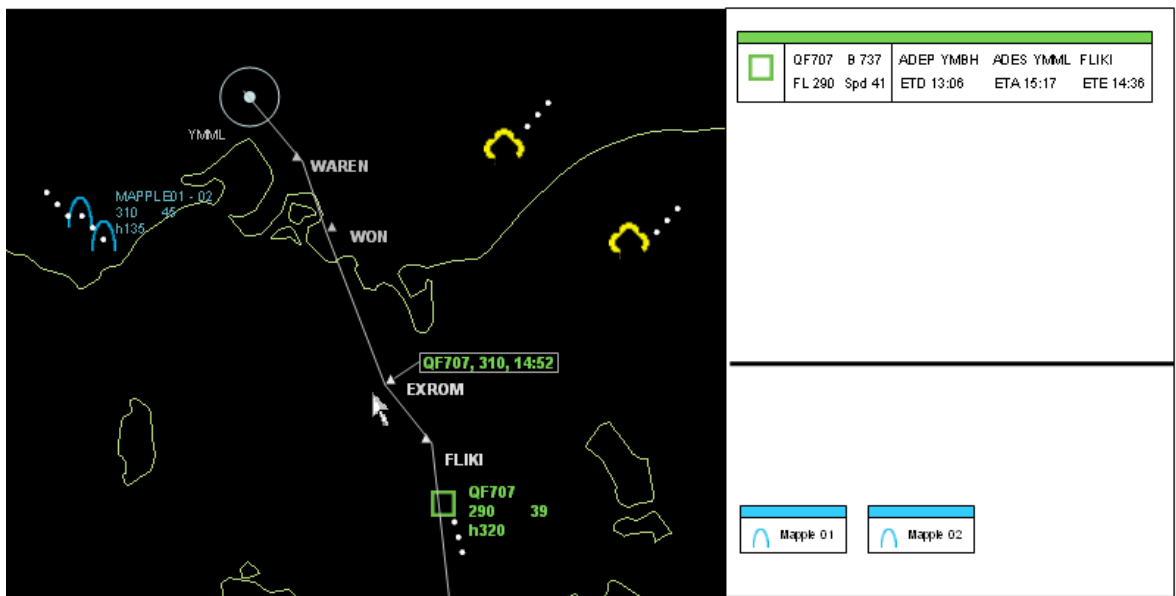


Figure 3. GUI snapshot

provided to support them in performing their task and to help them understand the situation. So, it should not be too intrusive for the operators. Second, the information is provided in a workspace (as mentioned above) to enable information consultation without hiding the situation on display. This workspace should help de-clutter the display while providing easy access to additional information. The integration of new information within the display takes place when operators open the window to consult the information. At this point, the system queries the GUI state to understand the current situation and decides what needs to be changed on the display to contextualize the information to be delivered. For now, only simple rules have been implemented to clear the working space and make room for the incoming information. Those rules consist of minimizing any strip window that has the neutral or friendly status and putting them in the passive workspace (bottom part of the working space). Those rules include also resizing the active workspace by introducing a scrollbar and showing only the top part of the active workspace (which should correspond to the most recent information delivered).

5. CONCLUSION

Future AEW&C operators will deal with a large amount of information. At any point in time, however, they do not necessarily need access to all available information. We propose to aid the operators to focus on important information and thus make better decisions by displaying only crucial data and highlighting its importance.

To do so, we use a delivery engine (the VDP) that we have integrated with the operators' GUI. The VDP collects and organizes the information that operators need to perform their task, and designs the final display. Then, the GUI delivers the information in the context of the operators' activities. Thus, the system grounds the delivery of information within the context of the activities the operator is performing. The GUI becomes task sensitive and intelligent, by proactively delivering appropriate information tailored to the current task, and providing smooth transitions between displays. It adds flexibility to the GUI because of the GUI's capability to dynamically interpret declarative specifications related to the display of the data and the interaction process.

6. ACKNOWLEDGMENTS

We wish to thank Robert Tot for his work on the task analysis and modeling and the surveillance operators and fighter controllers from the RAAF who participated in the task analysis interviews. We acknowledge the support of Boeing (Research Agreement WT-SIDA 010.2) and CSIRO.

7. REFERENCES

[1] André, E. and Rist, T. (1995). Generating coherent presentations employing textual and Visual material. In *Artificial Intelligence Review*, Special volume on the Integration of Natural Language and Vision Processing, 9 (2/3). 147-165.

[2] Budzik, J., Hammond, K.J. and Birnbaum, L. (2001). Information access in context. In *Knowledge-Based Systems*, 14, 37-53, Elsevier.

[3] Colineau, N. and Wan, S. (2001). Mobile delivery of customized information using Natural Language Generation. In *Monitor (Special Issue on Wireless Communication Special)*, 26(3), Sept.-Nov. 2001, 27-31.

[4] Colineau, N. and Paris, C. (2003). Task-Driven Information Presentation. In *Proc. of OZCHI'03*, Brisbane, Australia, Nov 25-28.

[5] De Carolis, B., De Rosi, F., Andreoli, C., Cavallo, V. and De Cicco, M.L. (1998). The dynamic generation of hypertext presentations of medical guidelines. In *The New Review of Hypermedia and Multimedia*, 4, 67-88.

[6] Fekete, J-D. and Plaisant, C. (1999). Excentric labeling: dynamic neighborhood labeling for data visualization. In *Proc. of CHI'99*, Pittsburgh, PA, May 15-20, 1999, 512-519.

[7] Geddes, N. and Hammer, J. (1991). Automatic display management using dynamic plans and events. In *Proc. of the 6th Symposium on aviation psychology*. 90-95.

[8] Horvitz, E., Jacobs, A. and Hove, D. (1999). Attention-Sensitive Alerting. In *Proc. of UAI '99*, Stockholm, Sweden, July 1999, 305-313.

[9] Maglio, P., Baret, R., Campbell, C. and Selker, T. (2000). SUITOR: An attentive information system. In *Proc. of IUI'00*, New Orleans, Louisiana.

[10] Mann, W. and Thompson S. (1988), Rhetorical Structure Theory: Towards a Functional Theory of Text Organization. In *Text*, 8(3), 243-281, 1988.

[11] McKeown, K. R., Chang, S-F., Cimino, J., Feiner, S. K., Friedman, C., Gravano, L., Hatzivassiloglou, V., Johnson, S., Jordan, D. A., Klavans, J., Kushniruk, A., Patel, V. and Teufel, S. (2001). PERCIVAL, a system for personalized search and summarization over multimedia healthcare information. In *Proc. of JCDL '01*, June 24-28, Roanoke, Virginia.

[12] Moore, J. and Paris, C (1993), Planning text for advisory dialogues: Capturing intentional and rhetorical information. In *Computational Linguistics*, 19(4), 651-694.

[13] Rutledge, R., Bailey, B., van Ossenbruggen, J., Hardman, L. and Geurts, J. (2000). Generating Presentation Constraints from Rhetorical Structure. In *Proc. of the 11th ACM Conference on Hypertext and Hypermedia*, San Antonio, Texas, May 30 – June 3, 19-28.

[14] Shalin, V. and Geddes, N. (1994). Task dependent information management in a dynamic environment: concept and measurement issues. In *Proc. of the IEEE International Conference on Systems, Man and Cybernetics*. Vol. 3. 2102-2107.

[15] Yunwen, Y. and Fischer, G. (2002). Information delivery in support of learning reusable software components on demand. In *Proc. of IUI'02*, San Francisco, CA, January 13-16.

[16] Zhou, M. and Feiner, S. (1998). Visual task characterization for automated visual discourse synthesis. In *Proc. of CHI'98*, Los Angeles, CA, April 18-23, 392-399.