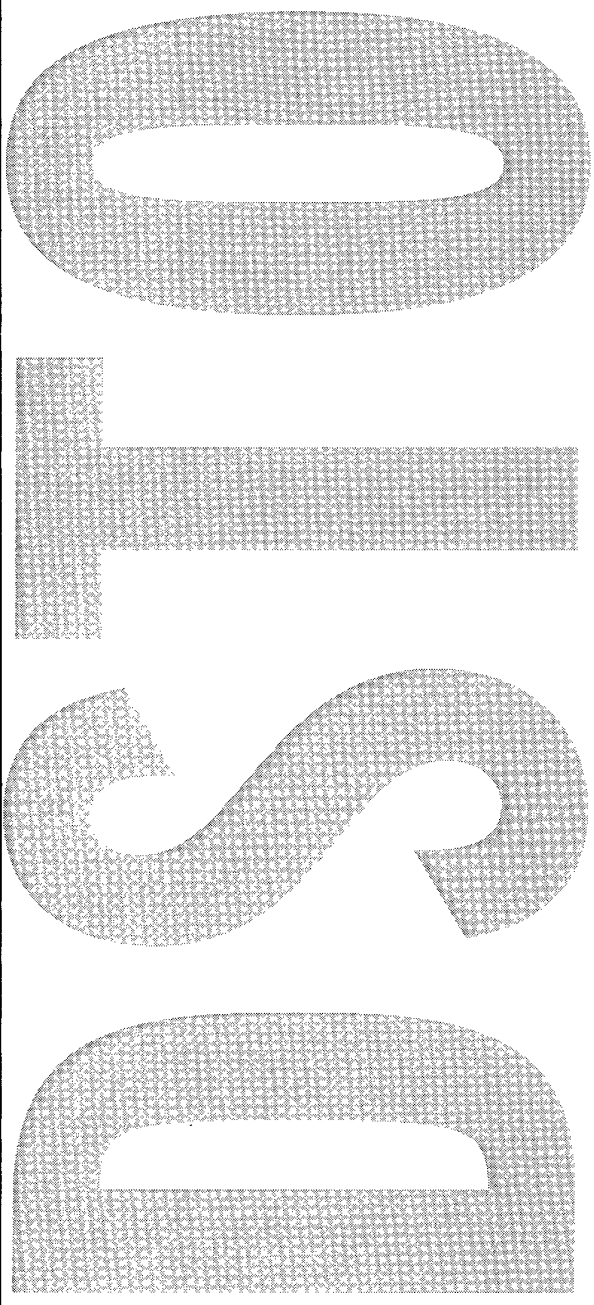
DEPARTMENT OF DEFENCE
DEFENCE SCIENCE & TECHNOLOGY ORGANISATION
DSTO



Information Visualisation using Composable Layouts and Visual Sets

Tim Pattison, Rudi Vernik and Matthew Phillips

DSTO-RR-0216

Information Visualisation using Composable Layouts and Visual Sets

Tim Pattison, Rudi Vernik & Matthew Phillips

Information Technology Division
Electronics and Surveillance Research Laboratory

DSTO–RR–0216

ABSTRACT

A modern military enterprise is characterised not only by its people, physical infrastructure and geography, but also by its business processes, knowledge management practices and fluid organisational structures. Management, coordination, planning and development of the enterprise all require awareness of its current state. To aid these functions, the DSTO task JNT 00/130 entitled "Assembly and Deployment of Defence Visualisation Solutions" (ADDVIS) proposes the use of information visualisation techniques to produce integrated enterprise situation awareness pictures tailored to meet the requirements of ADF functions such as capability development, system management and self-synchronisation. To this end, Information Technology Division (ITD) is performing research and development (R&D) into the next generation of information visualisation systems which will enable the rapid assembly and deployment of Defence visualisation solutions. InVision is a component-based software architecture for the rapid prototyping of information visualisation solutions. Its evolutionary implementation has given rise to an experimental component infrastructure with the aid of which the assumptions and goals of the ADDVIS task are being tested. This report describes the CLOVIS class of views, along with the associated supporting InVision infrastructure. The versatility and generality of the CLOVIS class of views described in this report makes it ideal for rapid prototyping of information visualisations. Its subsumption of a number of existing information layouts constitutes significant progress towards one of the key goals of InVision: the integration of various visual representations, with each chosen on the basis of its particular suitability to the task at hand.

Published by

DSTO Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury, South Australia, Australia 5108

Telephone: *(08) 8259 5555*
Facsimile: *(08) 8259 6567*

AR No. AR-011-960
August, 2001

APPROVED FOR PUBLIC RELEASE

Information Visualisation using Composable Layouts and Visual Sets

EXECUTIVE SUMMARY

Computer-based visualisation has long been recognised for its ability to rapidly convey large amounts of information. Military situation displays, for example, can simultaneously show the type, size, location, readiness and mobility of multiple units within the relevant geographic context. Three dimensional visualisations of the results of simulations which exercise proposed platforms, equipment or force structures can be used to inform and justify capability development decisions. And visualisations of systems such as computer, road or power distribution networks allow the intuitive display and rapid assimilation of potentially voluminous system status and performance information.

Information visualisation concerns the visualisation of abstract information which either has no visible embodiment in the real world, such as server loading in the above computer network, or for which strict adherence to that embodiment may severely limit the amount or types of information which can be displayed. As an example of the latter category, consider the distortion of an "aerial" view of a wide-area computer network to reveal greater detail of the inter-connected local area networks, or such that proximity between nodes reflects traffic volumes rather than geographical distance.

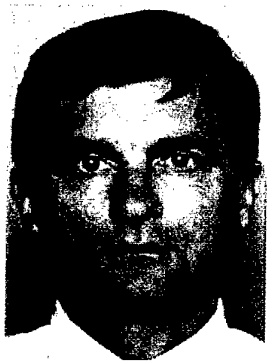
A modern enterprise – military or otherwise – is characterised not only by its people, physical infrastructure and geography, but also by its business processes, knowledge management practices and fluid organisational structures. Management, coordination, planning and development of the enterprise all require awareness of its current state. To aid these functions, the DSTO task JNT 00/130 entitled "Assembly and Deployment of Defence Visualisation Solutions" (ADDVIS) proposes the use of information visualisation techniques to produce integrated enterprise situation awareness pictures tailored to meet the requirements of ADF functions such as capability development, system management and self-synchronisation of the network-enabled enterprise.

To this end, Information Technology Division (ITD) is performing research and development (R&D) into the next generation of information visualisation systems which will enable the rapid assembly and deployment of Defence visualisation solutions. InVision is a component-based, knowledge-enabled software architecture for the rapid prototyping of information visualisation solutions. Its evolutionary implementation has given rise to an experimental component infrastructure with the aid of which the assumptions and goals of the ADDVIS task are being tested. This report describes the CLOVIS class of views, along with the associated supporting InVision infrastructure. The versatility and generality of the CLOVIS class of views described in this report makes it ideal for rapid prototyping of information visualisations. Its subsumption of a number of existing information layouts constitutes significant progress towards one of the key goals of InVision: the integration of various visual representations, with each chosen on the basis of its particular suitability to the task at hand.

Authors

Tim Pattison
ITD

Tim Pattison received a B.Sc. (Ma) in Applied Mathematics and Computing in 1988, a B.E. (Hons) in Electrical and Electronic Engineering in 1989 and a Ph.D. in neural network modelling of early vision in 1994, all from the University of Adelaide, South Australia. In 1999 he also completed a Graduate Certificate in Management (Scientific Leadership) through the University of South Australia. From 1993 to 1999, Tim worked in the Communications Division of the Defence Science and Technology Organisation (DSTO) on a variety of research topics including dual-satellite geolocation, loss-prioritised transport-layer protocols and social network analysis for the elicitation of communications requirements. He is now a Senior Research Scientist in the DSTO's Information Technology Division, where his technical interests include information visualisation systems, information retrieval, pervasive computing and software agents.



Rudi Vernik
ITD

Rudi Vernik is a Principal Research Scientist at the Defence Science and Technology Organisation. In his position as Head of Enterprise Visualisation, Instrumentation and Synchronisation Group, he leads research activities aimed at supporting Defence clients in their development of capabilities for knowledge and information-based warfare. Specific areas of interest include information visualization, component-based software engineering, enterprise instrumentation, and information domain modeling. Rudi has a PhD in Computer and Information Science (Software Engineering) from the University of South Australia. He also has a Bachelor of Electronics Engineering (with Distinction) and a Diploma of Communications Engineering from the Royal Melbourne Institute of Technology. He is a member of the IEEE.

Matthew Phillips
ITD

Matthew Phillips is a researcher employed in Enterprise Visualisation, Instrumentation and Synchronisation Group, Information Technology Division. His research interests include distributed systems, programming language design, software visualisation and object-oriented software engineering. Matthew has a Bachelor of Computer Science (with Honours) from The University of Adelaide.

Contents

1 Introduction

Graph visualisation and navigation techniques are known to assist the user in exploring large datasets which contain inherent relations among the data elements [1]. Examples of such data include world-wide web and intranet pages, computer and communication networks, software systems, social networks and distributed databases. Conventional graph-drawing techniques typically scale poorly with the number of vertices (data elements) and edges due to the increasing layout computation and decreasing resolution available on fixed-sized displays. However, if we remove the requirement that all information be displayed simultaneously, interactive techniques such as navigation, elision, filtering, searching, details on demand, and focus and context approaches, which are common in information visualisation [2], can be used to circumvent these limitations (see e.g. [3,4]).

Here we consider the use of graph drawing and information visualisation techniques for the flexible visualisation of information which can be modelled as an attributed graph. Despite its apparent simplicity, an attributed graph can model a wide range of different types of information, including, for example, system descriptions and database content. Information about the entities, to which the graph vertices correspond, and the relationships between them, to which the graph edges correspond, is modelled as vertex and edge attributes respectively. Although many graph drawing techniques exploit the information inherent in the graph edges, graph visualisation tools provide little or no support for the visualisation or exploitation of vertex and edge attributes, beyond perhaps the use of simple text labels. Furthermore, graph drawing techniques often produce uninformative or unaesthetic results when applied to sparse graphs, or to those lacking edges entirely. Yet data in the latter category can be richly exploited by many scientific and business data visualisation tools. Vertex attributes are also the focus of database visualisation tools [5,6,7], although some tools in this class also include graph visualisation techniques (see e.g. [8]). In short, no single class of information visualisation tools or techniques provides comprehensive support for the visualisation of information modelled as an attributed graph.

In this paper, we present the Composable Layouts and Visual Sets (CLoViS) framework for the visualisation of information which can be modelled as an attributed graph. The framework provides for the integration of a broad range of applicable information visualisation techniques through the flexible composition of their layouts and the consistent treatment of information overlays for the visualisation of attributes. The user is thereby provided with the freedom to simultaneously exploit the strengths of a number of information visualisation techniques. *Any* graph, database, scientific data or other visualisation technique whose underlying visual representation can be viewed as a layout of the vertices of an attributed graph can be accommodated by the framework, regardless of how much or how little of the information contained in the graph is actually exploited by the layout. This unifying model will be seen in Section 3.2.2 to suggest entire classes of potentially useful layout techniques which have yet to be considered in the literature.

This paper is structured as follows. In Section 2, the information visualisation concepts relevant to the flexible visualisation of information modelled as an attributed graph are reviewed. In Section 3, the CLovis class of views and supporting software framework are described. In Section 4, examples of the preliminary validation of CLovis view concepts

in several application domains are presented. In Section 5, various aspects of the CLOVIS framework are analysed, future features suggested, and topics of future research identified. And finally, the original contributions of this paper are summarised in Section 6.

2 Flexible visualisation of attributed graphs

2.1 Attributed graphs

Datasets which can be modelled as multi-attributed directed graphs arise for example from binary entity-relationship databases [9]. The visualisation of such data presents both opportunities and challenges for conventional graph drawing approaches. Most obviously, the presence of edge and vertex attributes increases the amount, types, and possible combinations, of information which may need to be represented, while at the same time offering the means by which unwanted graph elements can be filtered out. In addition, edge attributes can be used to select a subset of the edges for use in vertex layout, while vertex attributes can be used to layout vertices by applying multidimensional scaling [10] to some measure of their similarities, or mapped directly into vertex positions (see e.g. [11, 3, 7]). Attributes can also serve as the basis for preferential treatment when attempting to minimise aesthetic imperfections such as vertex occlusions or edge crossings. In short, the presence of attributes on graph elements opens up many research questions in graph visualisation which are not addressed in the graph drawing literature. The CLOVIS framework presented in this paper facilitates the exploration of this problem space.

2.2 Efficient traversal

The visual traversal of large graph-based data structures can be made more efficient by super-imposing a viewing structure which has efficient traversal properties [12]. One such structure is the (ideally balanced) tree, which can be super-imposed on a graph by hierarchically clustering the nodes. Traversal of the data can then be performed through navigation of the corresponding cluster tree. The use of a separate view for navigation of this cluster tree is undesirable in that it both consumes valuable display real-estate and requires the shifting of attention between it and the view of the original graph. A conventional drawing of the cluster tree in the same view as the original graph would tend to defeat the goal of uncluttering the view. This dilemma can be resolved through the use of a Tree-Map [13] in which each rectangle, corresponding to a non-leaf node of the cluster tree, serves as a container for the display of not only the corresponding sub-clusters, but also any leaf nodes, representing the vertices of the original graph. In addition to the inherent space efficiency of the Tree-Map, elision of the content of clusters and the interactive choice of a sub-tree for display could be used to reduce the complexity of the display, as well as the computation required for layout. Although the Tree-Map was originally conceived as a summary view for an entire tree, the use of a Nested Tree-Map [13] would allow the user access to internal nodes of the tree for the purposes of sub-tree exploration and elision.

2.3 Generalised layout

The Tree-Map uses an alternating pattern of horizontal and vertical linear layouts of rectangular containers. To improve the flexibility and customisability of the resultant display, at the expense of some space efficiency[1], we propose here a generalisation of this approach in which: a greater choice of intra-container layouts, including graph layouts, is available; each container may use a different layout; and a greater choice of shapes for leaf nodes is offered. Given a greater choice of intra-container layouts, those based on the graph structure can be supplemented by those – such as scatterplots – which reflect the attributes of the graph vertices or edges. In addition, both structure and attributes might be ignored in the interest of computational or space efficiency, especially in the early stages of exploratory interaction with large data sets. In cases where structure-based graph layouts are still appropriate, the goals of interactivity and computational scalability are also assisted by the divide-and-conquer approach inherent in the use of per-container layouts, since each such layout is typically applied to a small sub-graph of the original graph. The use of distinctive layouts for each container can provide a pre-attentive visual cue to the distinction between containers, which is in addition to container indentation and spacing. In some application domains, there may also be "natural" or conventional layouts for certain object types, such as a hierarchy tree for the members of a department, a two-dimensional array for image pixels, or a linear (flow) layout for the members of an ordered set. Similarly, the use of readily-recognised shapes for the vertices of the original graph can aid rapid comprehension, especially in application domains for which there is a well-established symbology.

The CLOVIS framework exploits these three generalisations to provide a powerful new class of views based on the flexible composition of vertex layouts. The combinatorial variety of views in this class will benefit both the exploratory data analyst, who values flexibility in pursuit of an evolving understanding of the data, and the operational user, for whom a more tailored view can now be produced. Support for composable layouts is provided by the Layout Composition Framework (LCF), which is described in Section 3. The LCF is a component-based software framework for flexible layout composition in which each layout is implemented as a pluggable component. The design and implementation of this framework adapts and extends standard techniques from the field of software user interface design (see e.g. [14]), where the choice and composition of layouts for user interface components is commonplace. In contrast, layout composition does not appear to have been widely adopted in the graph drawing literature, perhaps because of the need for the *a priori* specification of the combination of layouts to be used. Nevertheless, the drawing of combinatorial data structures using the composition of tree, circular, spiral and linear layouts was supported by the *Adocs* system [15], while tree visualisation using per-subtree layouts is supported by the Graph Visualisation Framework [16].

2.4 Clustering

Vertex clustering involves the allocation of the vertices of a graph to sets on the basis of pre-specified similarity criteria. These criteria may take account of graph structure or

[1]The reduction in space efficiency can be countered by the use of tree navigation techniques.

vertex attributes [1], spatial proximity in a given layout [3], or even information not represented in the graph [1]. Once the clustering criteria have been specified, a corresponding clustering algorithm can usually be designed and automated.

A *hierarchically clustered* graph is one in which the vertices are assigned to sets – or *clusters* – which form a recursive partitioning of the set of vertices of the graph. Although algorithms exist for drawing hierarchically clustered graphs [17], they do not address the problems of visual clutter and computational cost for large graphs, or provide for efficient interactive traversal of the graph. Vertex clustering has been used to reduce the visual complexity and layout cost of graph drawings [18, 4] by reducing the number of vertices, and hence also edges, to be displayed. The elision of cluster content used to achieve this reduction is also amenable to interactive use, allowing drill-down and roll-up through the tree representing the cluster hierarchy (see e.g. [19]). Elision of the content of a cluster should result in the aggregation or *abridgement* [19] of the edges incident on the vertices in the cluster, and the association of these aggregate edges with the meta-node – or in our case container – corresponding to the abridged cluster. The container should then be treated as any other vertex by a graph layout algorithm, which sees only these abridged edges. Generic extension of the concept of abridgement to the case of attributed graphs will require the specification of an appropriate strategy(s) for the fusion of edge and vertex attributes.

Although hierarchical clustering serves as a convenient starting point for explanation of our approach, the tree in question should properly be thought of as describing the container nesting pattern used for layout composition, rather than a cluster tree. While the user is free to specify (clustering criteria which result in) an hierarchical clustering of the vertex set, there is no requirement that it be hierarchical, nor that the clusters be mutually exclusive.

2.5 Knowledge Crystallisation

The users of an information visualisation system can be caricatured as falling into two classes [5]: the operational or executive user, who interacts with largely pre-defined views to answer routine questions; and the exploratory data analyst (EDA), whose views must be continuously tailored to support an emerging understanding of the data. In practice, however, there is a continuum of user types between these two extremes. Visualisation tools should be consciously positioned to support part or parts of this continuum. Our prototype software framework for CLOVIS views has focused initially on the EDA end of the spectrum. Support for the executive user will be added incrementally, with a simplified user interface hiding low-level functionality and supporting macros and intelligent user agents.

An important application of information visualisation at the EDA end of the spectrum is knowledge crystallisation. In a knowledge crystallisation task, "[...] a person gathers information for some purpose, makes sense of it [...] by constructing a representational framework [...], and then packages it into some form for communication or action." [2]. Having identified information which may be of interest, and obtained it in the form of an attributed graph, the user must find a representational framework which is appropriate to the task at hand. Note that although its organisation into an attributed graph already

constitutes a representational framework, which may even be supplemented by the schema of the database (if any) from which the data are sourced, neither may be well-suited to the current task.

Vertex clustering is one of a number of tools which can be used in the search for a representational framework. A suitable choice of clustering criteria is usually arrived at through an iterative process during the sense-making stage, rather than being evident *a priori*. Clustering support for knowledge crystallisation tasks should therefore provide for the interactive selection of the clustering algorithm and the specification of its parameters. The LCF currently supports a flexible querying approach to clustering, in which a cluster is specified on the basis of an interactively specified graph template which must be matched by each of its members. This template takes into account vertex and edge attributes, as well as graph structure. In the future, other automated clustering approaches should also be supported by the LCF.

Visual communication of the results of the sense-making process relies on the construction of one or more views which make explicit any relevant patterns or other information identified in the data. Identification of regularities in the data allows for the simplification of the problem space in support of decision-making. The cognitive load on the decision-maker can also be reduced by making explicit any information which is relevant to the decision. For example, if the path distance between vertices through the attributed graph is significant, then a suitable visualisation might be a multi-dimensional scaled representation of path distance [20], in which the Euclidean distance between vertices approximates the corresponding path distance. By the nature of the knowledge crystallisation process, the patterns or other relevant information, or the criteria for finding them, cannot be specified *a priori*. Similarly, it is not possible to choose in advance the relative priority of various aesthetic criteria for the drawing of a relevant sub-graph. Graph-based support for knowledge crystallisation should therefore support experimentation with a range of vertex layout algorithms through their interactive selection and customisation.

2.6 Visual sets

As part of the sense-making process, the exploratory data analyst engaged in knowledge crystallisation will typically need to query the data stored in the attributed graph. The querying mechanism might involve the stipulation of criteria which must be matched by a specified vertex or edge attribute, and return the set of matched vertices or edges. Alternatively, it might re-use the template-based query mechanism – currently used by the LCF in the assignment of vertices to containers – to match larger subgraphs. In order to support the visualisation of the query results, while preserving the user's existing mental map of the data and avoiding the need to shift attention to a different view, the CLOVIS framework provides for the assignment of visual attributes to the currently-displayed graph elements selected by the query. Other than position, *any* visual attribute, including size, shape, colour, texture, visibility and label font attributes, can be specified either explicitly or implicitly. The implicit specification of visual attributes, through the specification of a mapping of the selected graph attribute values onto visual appearance, has been demonstrated in information visualisation environments such as Datasplash [6, 7] and IVEE [21]. A set of graph elements and an associated specification of their visual

appearance is referred to here as a *visual set.* In contrast, a *collection* in *Visage* [5] corresponds to a set of graph elements without an associated appearance specification.

The concept of visual sets extends the notion of transparencies which are overlaid on maps, radarscopes and other displays in two important regards. Firstly, visual sets can specify visual attributes such as size, shape and visibility which cannot be easily and independently changed by super-positioning alone. And secondly, they permit finer control over the combination of multiple "overlays". Whereas multiple overlays affecting the same visual attribute on the same graph element would typically produce unappealing results, visual sets allow more flexible arbitration of such conflicts, through for example the use of a stacking order to determine which "overlay" should prevail. Furthermore, the accumulation of overlays does not, for example, permit the selective highlighting of the intersection set of the graph elements to which they are applied, which could help to reduce visual clutter. In contrast, visual sets open the way for arbitrary set-theoretic combinations of the underlying graph element sets. Despite these differences, however, there are a number of similarities in the way overlays and visual sets are managed. For example, both can be removed to control visual clutter and later re-applied, or grouped to form composites. To facilitate these and other manipulations, both should also be named and hierarchically organised to facilitate easy identification and access.

The generalisation of overlays to apply to attributes such as size and shape comes at a cost. Since changing these attributes can reduce the quality of the current vertex layout by for example introducing graph element occlusion, it may be necessary to rerun the vertex layout algorithm(s) after one or more visual sets is activated. Animation techniques described in [22] are used to preserve the user's mental map during the re-layout process by allowing them to visually track the movement of each node.

3 Composable Layouts and Visual Sets (CLOVIS)

In this section, we describe the use of *composable layouts* and *visual sets* (CLOVIS) for the visualisation of abstract information which can be modelled as an attributed graph. The imposition of a container tree as a viewing structure on the attributed graph provides for efficient navigation of the information it contains, as required by the operational user. On the other hand, the combinatorial variety of vertex-container assignments, per-container layouts, visual set memberships and associated appearances available for the visualisation of a single graph offers the high degree of flexibility demanded by the exploratory data analyst (EDA). Of course, the EDA will in general also require access to, and coordination between, a variety of non-CLOVIS visualisations, including such staples as charts and tables. The integration of these various view types into visualisation solutions which can be rapidly assembled and deployed is the subject of the *InVision* project [23], a brief overview of which is provided in Section 4.1.

Prototype component-based software infrastructure has been developed within the InVision framework to support the creation and exploration of CLOVIS visualisations. Its design emphasises support for the flexible definition and customisation of views rather than the optimal design of any individual view, since suitable optimality criteria are often not known in advance. The requirement for interactive information visualisation, as opposed to

batch-mode graph drawing, has also lead to an emphasis on low computational complexity.

Section 3.1 describes the underlying attributed graph model to be visualised. The Layout Composition Framework and visual set management tools provided by the CLOVIS infrastructure are described in Sections 3.2 and 3.3 respectively.

3.1 Attributed graph model

Each vertex of the attributed graph represents a modelled entity, and each directed edge a relationship between entities. Information about the entities and their relationships is stored as attributes on the corresponding vertices and edges. Each attribute is typed to facilitate sorting, querying, fusion and other operations on the graph elements. Each graph element – vertex or edge – can have any number of attributes of various types. The only mandatory attribute is the meta-name, whose value controls the set of (other) attributes allocated to that graph element. The unique role of this attribute has lead some authors to distinguish it with the title "label" [24], and hence to refer to the graph as a *labelled* attributed graph.

The data represented by this model could arise for example from a binary entity-relationship database (see e.g. [9]). Accordingly, the attributed graph has an associated meta-graph – the equivalent of a database schema – in which a single vertex (respectively edge) represents the set of graph vertices (respectively edges) bearing its label as their meta-name attribute. This meta-graph, which includes the specification of attribute names and types, along with bounds on the number of attributes and edges, constitutes a set of constraints on the underlying attributed graph which can be used both to provide for its efficient storage and to validate its structure and content. Subgraphs of the meta-graph, with attribute values specifying regular expressions which must be matched by the corresponding graph element attributes, also serve as query templates to be matched by the graph.

A modelling framework which supports the creation, manipulation and querying of an attributed graph has been developed to underpin the CLOVIS component infrastructure. For both generality and consistency with other meta modelling languages, such as the Unified Modelling Language (UML) [25], this framework also supports additional features such as meta-element inheritance, and edges incident on other edges. Discussion of these features is however beyond the scope of this paper.

3.2 Layout Composition Framework

The Layout Composition Framework (LCF) is a component-based software framework which supports the creation and modification of highly flexible and customisable layouts of an attributed graph. Composition of layouts is achieved through: the specification of a hierarchy of containers which are to be displayed nested, as in a Tree-Map [13]; the allocation of a layout strategy to each container; and the allocation of graph vertices to containers. These three tasks are performed during the sense-making process of knowledge crystalllisation using the layout composition specification editor, which will be discussed shortly. Container indentation, as used in a Nested Tree-Map, ensures direct manipulation

access to the internal nodes of this hierarchy, which will be exploited for navigation and elision of the container hierarchy. Although container indentation results in a modest loss of screen real estate available for the layout of graph vertices, this loss is expected to be more than compensated by the increased freedom to choose what is displayed.

3.2.1 Layout composition tree

The hierarchy of containers can be modelled as a *layout composition tree*, whose leaf nodes correspond to visible graph vertices and whose non-leaf nodes correspond to containers. In order to simplify both the implementation and discussion of this tree, we note that a visible graph vertex can be thought of as an empty, visible container with a "null" layout for its (non-existent) contents. Each node of the tree specifies a *layout rule*, which dictates the appearance of the corresponding container and the layout of its contents. In order to avoid having to individually specify identical layout rules for a set of sibling containers which are to be assigned the same appearance and layout, a suitable shorthand notation is warranted. The LCF provides this shorthand for the case where a container is to be generated for each vertex in the result set of a query, specified by a *selection expression*, on the attributed graph. For this purpose, each layout rule is augmented with a selection expression, the result set of which can be optionally fused into a single, synthetic vertex if only a single container is required.

To illustrate the application of this shorthand notation, consider the case of an attributed graph which models the file system of a computer, and has vertices with the meta-names "file" and "directory". A single layout rule with a selection expression based on the "sub-directory" edges from the "directory" vertices can be used to generate containers with identical layout and appearance corresponding to the subdirectories of a specified directory. In a similar model which contains only "file" vertices, a single container for the contents of the directory of interest could be created using a layout rule with a selection expression based on the "pathname" attribute of these "file" vertices. Here the fusion option would be chosen in order to produce a single container. Elision of the contents of this directory would be achieved by failing to specify any subordinate layout rules.

Since the selection set of a parent layout rule can be copied to its children at no cost, the use of selection expressions in *all* layout rules neither imposes unnecessary query evaluation effort nor restricts the range of layouts which can be specified. It does, however, provide the means by which the efficiency of query evaluation, and the economy of specification effort, can be increased, by allowing the user to specify shared partial queries once in an ancestor layout rule, rather than multiple times in its descendants. Evaluation efficiency is also increased by explicitly anchoring one or more meta-vertices of the selection expression in the selection set of the parent layout rule, since the number of potential matches for each anchored vertex is reduced by the ratio of the size of the selection set to the total number of vertices in the attributed graph. If none of the anchored meta-vertices matches the selection set, then the result set must be empty and evaluation need proceed no further.

The layout composition specification editor shown in Figure 1 consists of the layout composition tree (left) and the *layout rule customiser* (right). Each node in the tree contains a layout rule for the corresponding container, the details of which are specified in the right-hand panel of Figure 1. The *layout strategy* for the selected layout rule is chosen

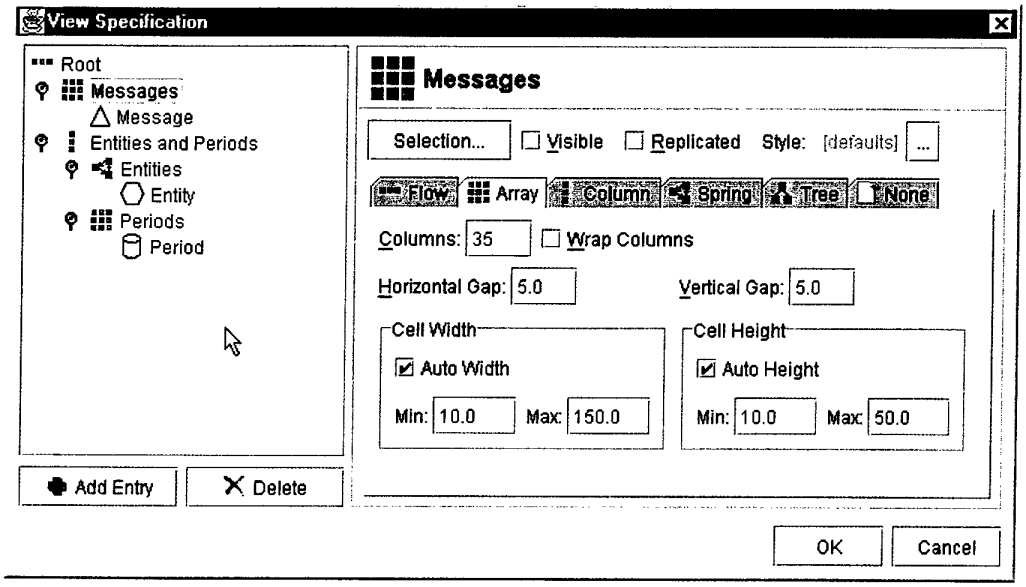


Figure 1: Layout composition specification window.

by selecting the appropriate tab in the layout rule customiser, and its parameters specified using the layout strategy customiser in the corresponding tab pane. In the example in Figure 1, the details of the "Messages" rule, corresponding to the array layout of triangular objects in the CLOVIS view of Figure 2, are shown. An array layout strategy has been selected, and the array width and vertex spacing specified. The interested reader is referred to Section 4.3 for an explanation of the content and application of the CLOVIS view in Figure 2. Other application examples are also included in Section 4.

The "Selection" button in Figure 1 invokes the selection expression editor shown in Figure 3, which facilitates the construction, by direct manipulation, of a subgraph of the meta-graph to be used as a query template. The vertices, edges and attributes of the meta-graph listed in the left-hand panel of Figure 3 can be dragged and dropped into the selection expression canvas on the right. The selection expression is matched against the underlying attributed graph, and for each match, the graph vertices corresponding to the checked meta-vertices are placed in the selection set of the current layout rule. In this example, the selection expression will match vertices having the meta-name "message", the "channel" attribute "VOICE", "DATA", "DICE_Link" or "To Controller", and which are connected to vertices having the meta-name "root" by a directed edge having the meta-name "contains".

The layout composition specification dictates a clustering of a subset of the vertex set which, despite its tree structure, need not be hierarchical. Sibling containers are not required to make mutually exclusive selections, and a container is not required to select a subset of its parent's selection. One consequence of these relaxations is that the same vertex can be represented in multiple containers. In this case, vertex duplication is used to preserve the container hierarchy, and hence also the two-pass layout approach to be

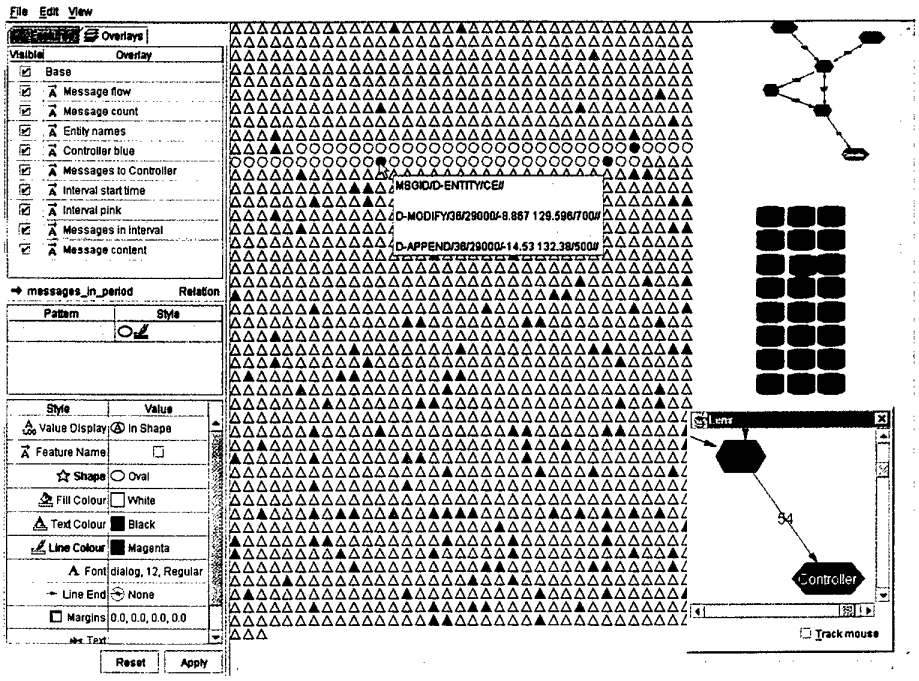


Figure 2: CLOVIS view of message traffic.

discussed shortly.

Since the user is free to define an hierarchical clustering if required, the set of possible CLOVIS views subsumes clustered graph layouts. The visualisation of a directory structure is an example of an hierarchical clustering in which the selection set for the directory container – a single "directory" vertex – is nevertheless not a superset of the selection set – consisting of "file" vertices – for the subordinate container which holds its non-directory contents. This freedom, to select any set of graph vertices which are reachable by an undirected path through the attributed graph from the selection set of the parent layout rule, can be exploited to produce non-hierarchical clusterings. Only in cases where the attributed graph is disconnected is this approach more restrictive than that used in Datasplash [7, 6], for example, where *any* vertex can be allocated to a container.

3.2.2 Layout strategies

The layout of the contents of a container is determined by a *layout strategy*, which is selected by choosing a tab in the layout rule customiser shown on the right of Figure 1. The term "layout strategy" acknowledges the "strategies" software pattern [14] which is common in user interface design. To illustrate the variety of layouts supported by the Layout Composition Framework, we classify existing layouts into one of three categories: graph, attribute or blind.

A *graph* layout reflects or is derived from the graph structure rather than the vertex or

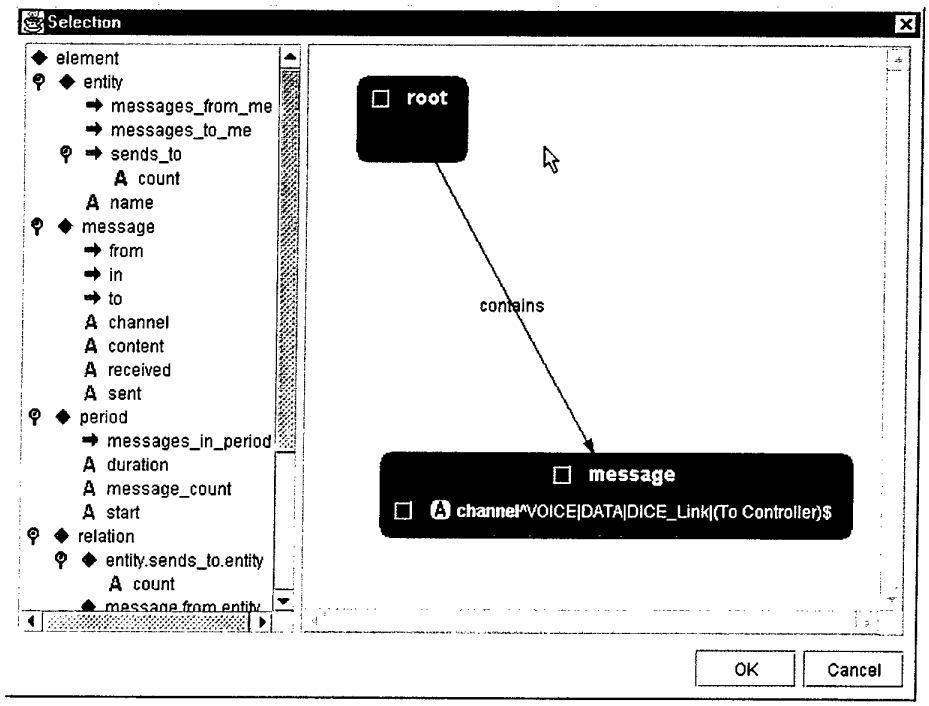


Figure 3: Selection expression editor window.

edge attributes. The LCF subsumes conventional graph drawings, since the user is free to define a single container with a graph layout involving all of the graph elements. However, the computational complexity of graph layouts (which may be $\mathcal{O}(n^2)$ or $\mathcal{O}(nlogn)$ for n vertices) is typically offset by the reduction in the size n of the vertex set to which it is applied. For example, the spring embedder [26] graph layout in Figure 2 is applied only to the red vertices, which correspond to the communicating entities in a simulation of a military messaging system. In the prototype CLOVIS implementation, graph layouts make use of the edges, if any, which are included by one or more currently-visible visual sets.

An *attribute* layout is based on the vertex attributes. A scatterplot (see e.g. [11]) without axes is an example of an attribute layout for numerical attributes. Similarly, ordinal attributes could be used to create an array in which not necessarily all cells are occupied by vertices of the original graph. An attribute layout typically has linear complexity, although more complex layouts in this class are also possible, such as those based on multidimensional scaling (see e.g. [10]) of a "distance" measure representing inter-vertex attribute dissimilarities.

A *blind* layout ignores both the structure and attributes of the attributed graph, and lays out the vertices according to a pre-determined but usually regular pattern, such as a (linear) flow, array or circular arrangement. For example, the alternation of horizontal and vertical flow layouts at successive levels of the layout composition tree produces the loose approximation to a Nested Tree-Map shown in Figure 4. This third layout category primarily includes layouts which have computational complexity linear in the number of

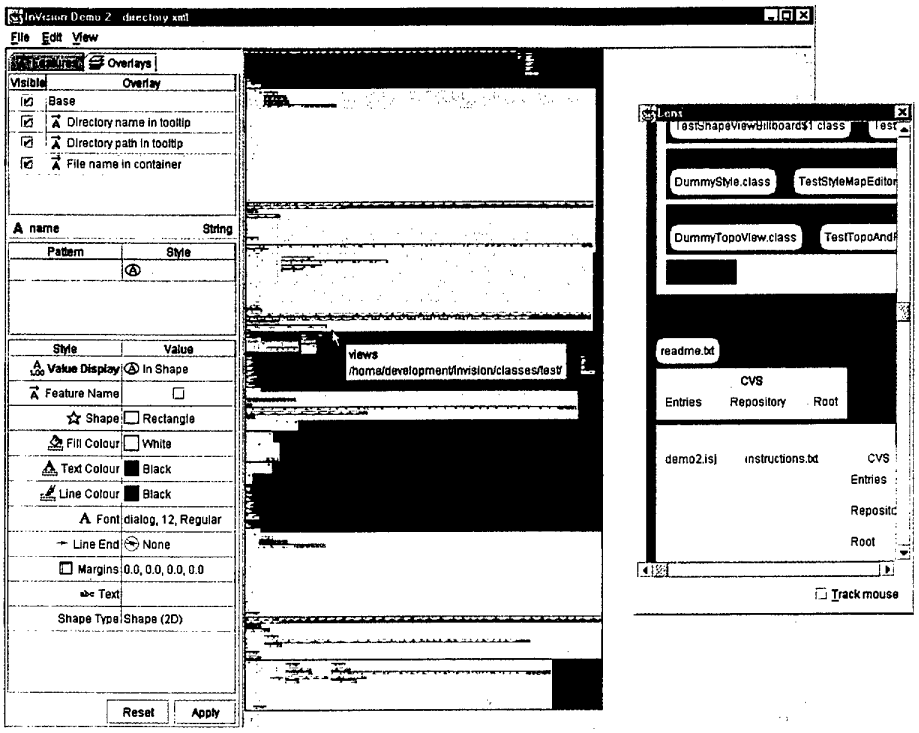


Figure 4: Simple, rapidly prototyped CLOVIS approximation to a Nested Tree-Map.

vertices to be laid out, and hence are more scalable than graph layouts. A random layout would also fall into this category. Many blind layouts also have linear spatial complexity (or inverse linear resolution), while still guaranteeing that vertex occlusion will not occur. This compactness is especially valuable for summary views of a large number of vertices, such as the triangular "message" vertices in Figure 2. The compactness of many blind layouts should be contrasted with the relatively inefficient use of space in conventional graph drawing, as bemoaned in [13]. Nevertheless, this category also includes some layouts with poor computational or spatial complexity. For example, a circumferential layout requires the use of suitable heuristics in order to give reasonable results in linear time (see e.g. [27, 26]), and has quadratic space complexity.

Distinguishing between the graph, attribute and blind layout categories has served to illustrate the variety of existing layouts which are currently or readily supported by the Layout Composition Framework (LCF). However, the LCF does not preclude the use of hybrids of these categories, or even layouts which fall outside of the categories. Entire classes of layouts not previously considered in the literature are therefore possible, such as graph layouts which prioritise the removal of edge crossings on the basis of edge attributes.

Each layout strategy is implemented in the LCF as a software component, which can be plugged in as required at either compile or run time. Each strategy has a corresponding tab in the layout composition specification window, and a layout strategy customiser in the associated tab pane, as shown in Figure 1. Consequently, the choice of layout strategy

for a given container is independent of that for all other containers. In the following subsection, we discuss how the actions performed by the layout strategies assigned to each container are coordinated to produce the resultant CLOVIS view.

3.2.3 Layout coordination

In the case of the Tree-Map, the relative area to be occupied by each tree node is pre-specified by the data, such as file size, which it represents. The generalisation of container layouts supported by the CLOVIS infrastructure necessitates an additional pass through the tree, during which containers and vertices bid for the space they require. Space allocation is then performed during the second pass, with containers having to "make do" with the space allocated to them. Further iterations of the bidding-allocation cycle could potentially result in more efficient use of the available space. For example, given insufficient space by the first round of allocations, a container might choose to elide its contents and revise its bid accordingly during a third pass; a fourth pass would then re-allocate the real-estate freed up by this elision. However, in order to retain interactivity when scaling up the LCF approach to large data sets, we have chosen not to pursue this option.

During the layout of a CLOVIS view, the layout strategy for each container is responsible for:

Pass 1 optionally asking its child containers for their preferred size given the specified maximum size of the root container, and reporting its own preferred size to its parent (if any)

Pass 2 laying out its own content, notifying its child containers of their allocated size and position, and triggering them to lay out their content.

The root container, corresponding to the root node of the layout composition tree, is the outermost container in a CLOVIS layout. Passes 1 and 2 are triggered, respectively, by asking the root container for its preferred size and, upon receiving the answer, triggering its layout. The maximum permissible size of the root container is specified in advance, and the minimum of this and its preferred size is used in the second pass.

Container layouts must be calculated in each pass – first to determine the preferred container size and then to perform the actual layout. For containers which are granted less than their preferred size, the area granted to them could be used as a window onto a larger underlying canvas, in the style of Datasplash *portals*. The current implementation of the LCF is based on two-dimensional views, although the nesting of containers with individualised layouts extends naturally to 3D. Each layout strategy should be responsible for its own extension to 3D, since while many layouts, such as the array, generalise readily to accommodate the third dimension, others, such as a text layout [28], may need special consideration.

3.3 Visual Sets

Each layout rule selects a subset of the graph vertices and, in addition to specifying a layout strategy, assigns an appearance to the corresponding container(s). The ability to assign a distinctive appearance to the result set of a visually formulated query on the attributed graph provides a potentially powerful mechanism for the visual exploration of the information it represents. The realisation of this potential requires the extension of this visual querying facility beyond the layout phase of the visualisation, and the inclusion of graph edges in query result sets. In CLOVIS visualisations, this extension is built on the use of visual sets.

As previously noted, a *visual set* is a set of vertices or edges of the attributed graph and a specification of their appearance. Membership of a visual set is specified implicitly through the use of a query on the attributed graph, with the option to constrain the result set in advance through direct manipulation selection. With the exception of position, any aspects of the appearance of a vertex or edge can be specified in any combination for a visual set.

Visual sets are defined in the CLOVIS prototype infrastructure using the Features panel shown on the left of Figure 5. The vertex or edge attribute on which membership of the visual set is to be based is selected from the tree view of the meta-graph in the top half of the panel. The mapping of this attribute to the appearance of the corresponding graph elements is then specified using the pattern and style specification tables shown in the middle and lower half of the "Features" panel respectively. Each row of the pattern specification table corresponds to a separate visual set, defining a pattern to be matched by the selected attribute for each graph element in the set, and the visual style to be assigned to the matched graph elements. If no pattern is specified, all elements bearing the selected attribute are affected by the corresponding style. If more than one row is specified in the pattern specification table, it is convenient for some purposes to treat the corresponding collection of visual sets as a single, compound visual set. The visual style to be applied to the members of a visual set is specified using the style specification table, and the resultant specification indicated in shorthand in the right-hand column of the pattern specification table.

The pattern and style specification tables together provide a flexible user interface for the exploration of the large range of possible mappings between attribute values and visual appearance. For example, a mapping of temperature to colour can be specified using a series of rows in the pattern specification table, each of which matches a different temperature range and assigns the corresponding colour. There is nevertheless considerable scope for the addition of shorthand mechanisms for the specification of such mappings, especially where numerical attributes are to be mapped onto finely-quantised values for visual attributes such as size or intensity.

Since multiple visual sets can be defined for the same view, the CLOVIS infrastructure provides a number of facilities which assist the user with their management. These facilities include allowing the user to specify descriptive names for each visual set, re-order the list, delete those which are no longer required, and turn each on and off. The left-hand "Overlays" panel in Figure 2 displays the list of overlays defined for the corresponding view on the right. Replacement of this list with a tree in a future extension of our pro-

totype CLOVIS implementation would facilitate the management and navigation of larger collections of visual sets, which would be organised hierarchically and labeled according to the line of enquiry being pursued. For example, the multiple visual sets defined in the above example to achieve the temperature-colour mapping would be grouped under a single parent node.

The process of constructing complex queries is rarely achieved in a single step. The exploratory data analyst will typically build increasingly sophisticated versions of one or more queries as their understanding of the data set evolves. Even in cases where the objective is clearly stated in advance, a user may wish to view intermediate results in order to ensure that the completed query will meet that objective. Visual sets offer this feedback in an easily-assimilated visual form. In Figure 2, for example, three separate visual sets have been applied to the orange triangles on the left of the view: a blue fill, a circular shape with magenta outline, and a tooltip. Members of the intersection set of the first two are readily apparent as blue hexagons with magenta outline. If this set is of interest, it can be targeted more selectively and highlighted more effectively by defining a new visual set which logically ANDs the two corresponding queries. Alternatively, the two visual sets could be combined set-theoretically, with the user specifying how the appearance of the resultant visual set should be derived from that of its constituents. Implementation of this option within the prototype CLOVIS framework would automate the otherwise manual process of re-specifying and logically combining the underlying queries. Of course, the combination of visual sets is not limited to the intersection set, which corresponds to the logical AND of the corresponding queries: any Boolean combination of queries corresponds to a suitable set-theoretic combination of their respective result sets.

In some applications it may be important to place the result of a complex query in the context of intermediate results. For example, Figure 2 shows an intersection set in the context of the two individual sets, allowing set size and membership comparisons. At first glance, there would seem to be a combinatorial space of possibilities for the simultaneous display of these intermediate results. However, several factors conspire to limit the amount of contextual information which can be effectively displayed. Firstly, two or more overlapping visual sets should not attempt to dictate conflicting settings for the same visual attribute. Secondly, some attributes are less salient than others to the pre-attentive vision, potentially requiring the user to resort to a visual search to identify set membership. And thirdly, the visual attributes available for use are not necessarily perceptually "orthogonal"; for example, the choice of colour can affect the perception of size.

In cases where the size, shape or visibility of graph vertices or edges changes as a result of the application of a visual set, the layout of a CLOVIS view may need to be updated at the user's discretion. Automated recognition of the need for a layout update, combined with the current animation of the layout changes resulting from the update, would assist the user in concentrating on the task at hand and preserving their mental map.

3.4 Summary

The flexibility of Composable Layout and Visual Set (CLOVIS) views arises from:

1. the large number of possible layout composition trees, including choices of

- allocation of vertices to containers
- container layout (layout type and parameter choice)
- layout composition tree structure
- edge sets for use in graph layouts

2. the large number of possible visual sets, including choices of
   - query-based set membership criteria
   - visual appearance

A description of each of these variables, and the interface through which they are specified, has been provided in this section.

The approach taken by the Layout Composition Framework to the reduction of computational and space complexity for the layout of attributed graphs has also been described. This approach involves a combination of divide-and-conquer, the use of compact, low-cost layouts where appropriate, an efficient two-pass layout algorithm, and elision of unwanted detail.

4 Applications

In this section, the versatility of the CLOVIS view family is demonstrated through its application to the visualisation of the InVision component architecture, a file system, message traffic, and importation relationships between Java classes.

4.1 InVision framework

Figure 5 provides an overview of the InVision component-based software architecture for the assembly and deployment of information visualisation solutions. The InVision architecture supports the integration and coordination of a variety of view types, including CLOVIS and chart views. Unlike other visualisation architectures with this design goal, however, InVision will also provide for their effective deployment as part of a visualisation solution through support for: the location and extraction of the information to be visualised from the information environment; interoperation with existing information technology tools and services; integration with user processes and organisational workflows; and user assistance.

The principal role of the CLOVIS view within the InVision framework is as a general-purpose view which is readily and iteratively customised to meet the user's specific visualisation requirements. Unlike statistical and graph visualisation techniques, respectively, it assumes neither numerical attributes nor the existence of graph edges, although it is able to exploit them when present. The view in Figure 5 was designed to meet the requirement of a visualisation software developer or integrator for an accurate and up-to-date architectural description of InVision which not only shows logical design structure but also implemented code elements. Additional detail such as class names can be added as needed using visual sets.

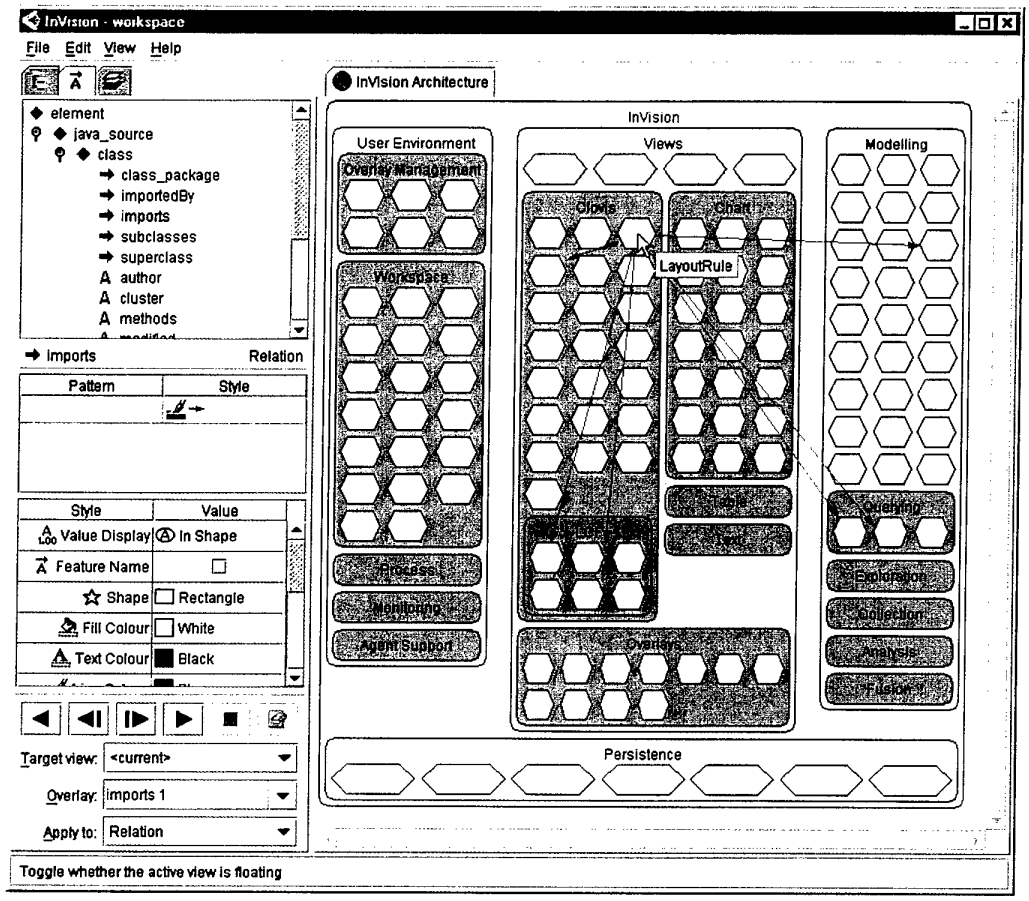


Figure 5: InVision component architecture.

Figure 5 shows a CLOVIS view of the attributed graph obtained by parsing the Java source code of the InVision component infrastructure for its package and class structure. The components and their composition are represented, respectively, by the containers and the pattern of their nesting. The containers, shown as rectangles with rounded corners, are colour-coded to facilitate the rapid appreciation and comparison of nesting depth. The classes, many of which are JavaBean components (see e.g. [29]), are shown as white hexagons. Their allocation to containers is achieved using selection expressions which filter on the "name" attribute of the "package" vertices to which they are attached by a "contains" relation. The directed importation relationships of the LayoutRule class are overlaid as a visual set.

The prototype CLOVIS infrastructure consists of the following InVision components in Figure 5. The Modelling framework supports the instantiation and management of the attributed graph to be visualised. Among its components is the Query framework, which provides for the specification and execution of queries on the graph. The CLOVIS framework, a component of the Views framework, includes the Layout Composition Frame-

work (LCF) described in Section 3.2, and the user interface for specification of the layout composition tree. The LCF incorporates layout coordination and plugin layout strategy components. Visual sets are implemented by the Overlay Management component of the User Environment framework.

The Views framework, is responsible for the coordination and integration of the various views supported by InVision. In addition to the CLOVIS view, a Chart view adapted from the *JChart* JavaBean is provided, while placeholders are shown for text, table and other views which will be implemented in the near future.

Within the User Environment framework, the Workspace framework provides support for user workspaces, which facilitate the organisation of views, data models, processes and user agents. The Process and Monitoring components will provide for the recording, playback and management of user interaction sequences involved in creating or interacting with views. An implementation of this functionality for the Visage database visualisation tool, and an associated user interface, are described in [30]. The Agent Support framework will allow the registration and invocation of agents which assist the user with the specification, refinement and interpretation of views and the location and extraction of the required information from the information environment.

Within the Modelling framework, the Exploration framework will provide for software agent exploration of the information environment, and the Collection framework for the extraction of that portion of the discovered information which is relevant to the user's visualisation requirements. Since the required information will in general need to be derived or distilled from the extracted information, information analysis and fusion frameworks are also slated.

4.2 Directory structure

Figure 4 shows a simple CLOVIS approximation to a Nested Tree Map of a file system, in which the leaf nodes corresponding to (non-directory) files are displayed as white, rounded rectangles. File and directory names, which are visible in the magnified lens view to the right, have been overlaid using a visual set. As in the Nested Tree Map, the nesting of containers, which use alternating horizontal and vertical flow layouts and are distinguished by different fill colours, indicates the corresponding nesting of directories. In contrast, however, the area occupied by a directory is dependent upon (although not directly proportional to) the total *number* of files and directories in the subtree rooted at that directory, rather than proportional to their total byte size.

Of course, more efficient use of the available space could be made by for example reducing the spacing between leaf nodes in the same container and using a layout strategy with more flexible wrapping. Nevertheless, the goal of simultaneous display of detail at all levels of an arbitrarily deep tree remains unattainable. Elision of directory content at the seventh level of nesting, distortion via a magnifying lens, and container navigation support are used here to circumvent this problem. More tailored Tree Map interfaces for browsing directory structure, such as that proposed in [31], might in future result from the use of the InVision infrastructure for rapid prototyping of the corresponding CLOVIS views.

4.3 Message traffic

Figure 2 shows the communicating entities, the messages exchanged between them, and each 100-second time interval during the simulation of a military messaging system. The communicating entities are represented as red hexagons, and are laid out using a spring-embedder graph layout based on the communication relations between them. In the magnified lens window, a message-count attribute can be seen overlaid on one of these edges. The direction of the magnified edge reveals that the "Pilot" entity sends messages to the "Controller", but receives none in return. The messages and simulation intervals are shown as orange triangles and green cylinders respectively, and are arranged in separate containers with array layouts. The messages sent to the blue "Controller" entity are assigned a corresponding blue fill by a visual set which selects *edges* with the meta-name "messages_from_me", and applies the specified visual style to the *vertices* on which they terminate. Similarly, the messages transmitted during the pink interval are shown as ovals with a pink outline. There are three messages (blue ovals) in the intersection of these two visual sets. The content of one of these messages is shown in a tooltip overlay.

4.4 Software structure

Figure 6 presents a CLOVIS view of the Java-based InVision software framework. The classes, shown as hexagons, are assigned to columns on the basis of the importation relations between them, such that if class A imports class B, then class B is either in the same column or to the right of A. The more highly imported classes are thereby "pushed" to the right. Classes belonging to one or more overlapping importation cycles are allocated to contiguous rows of the same column, and assigned an orange fill. The directed importation relations for the "IveWindow" class are overlaid in blue. One of these points to a class which is part of a cluster of overlapping cycles involving classes from the CLOVIS view software. The pattern of importation relations among the classes in this cluster is shown using a spring-embedder graph layout in a linked CLOVIS view. This view confirms the presence of multiple importation cycles, which form indivisible units from the point of view of code re-use, and identifies the "ClovisModel" class as central.

In past work involving the development and deployment of the *SeeADA* [32,33] software visualisation tool at a large Defence software contractor, the map-like, "coastal" outline and "topographic" features of this view provided convenient landmarks against which day-to-day changes in software structure were readily identified by the user.

4.5 Summary

In this section, the versatility of the CLOVIS view and of the underlying attributed graph information model has been demonstrated through a series of case studies showing its application in a variety of problem domains. The ease with which these example views were prepared through the user interface furthermore confirms the utility of the InVision infrastructure for the rapid prototyping of views in the CLOVIS view family.

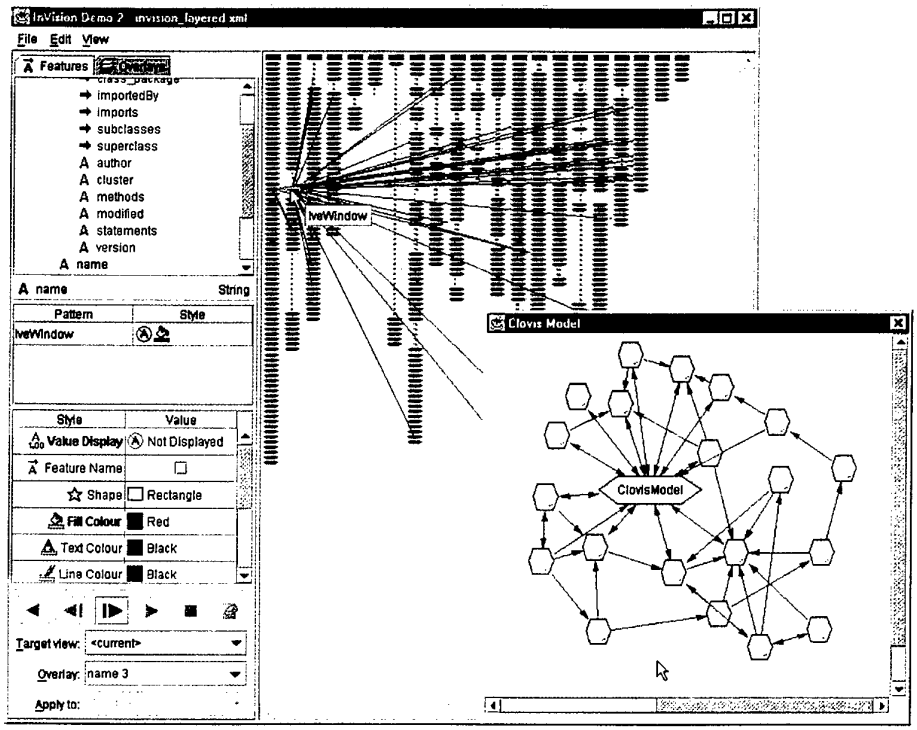


Figure 6: CLOVIS views of Java-based software.

5 Future Work

A number of areas of future investigation for CLOVIS views have been identified in previous sections. In this section, we survey some of the major challenges and research opportunities for the future development of CLOVIS views and the supporting framework.

Navigation through the container hierarchy is currently supported by the ability to zoom into a selected container while leaving sufficient surrounding context to be able to also select its parent. This might be supplemented by a host of other tree navigation techniques, including the ability to: elide the content of containers through direct manipulation (without changing the layout composition specification); jump directly back to the previously-zoomed container; and add containers to a list of bookmarks for future direct access.

Interactive elision of the contents of a container will in general require an update to the view layout. Layout updates can be optionally animated to help preserve the user's mental map. The animation component and the techniques it currently provides are described in [22]. At present, the animation deals only with the leaf nodes of the layout composition tree, being unaware of their surrounding containers. However, the appropriate choice of animation technique and its parameter settings will in general vary between containers. For example, a container having a graph layout whose goal is to maximise symmetry may require an animation technique which optimally preserves this symmetry during the tran-

sition. This goal may conflict with that of another container, which seeks to optimise and preserve the compactness of its layout. The facility to specify and customise per-container animation strategies would therefore seem appropriate. However, tying animation to containers raises a number of issues. For example, how would a vertex which moves *between* containers as a result of changing selection expressions be animated? Although this might be handled by the first common ancestor of the two containers, the layout of that ancestor currently knows nothing of the content of the two containers.

Clustering support in CLOVIS views is currently limited to the use of manually-specified queries for each container. These queries, and indeed the entire layout composition specification, could in future be generated automatically by a user-selected clustering algorithm which is applied to the attributed graph. The range of available clustering algorithms should extend beyond conventional structure-based graph clustering to attribute-based and hybrid clustering methods. A framework which allows the user or software developer to "plug in" the required clustering algorithms as components would assist with their management.

In the Layout Composition Framework, a container having an attribute or graph layout should be entitled to rely on its subordinate containers having attributes and relations respectively. In cases where a container corresponds to a single graph vertex, this will indeed be the case. However, if a container represents more than one vertex, then a strategy for fusing the corresponding vertex attributes, edges and edge attributes is required. This strategy may vary depending on the attribute type and the meaning of the attribute. It may for example be appropriate to take the mean value for a numerical attributes in some situations, and the maximum or concatenation in others. The specification of this fusion strategy should either form part of each layout rule, or be specified in the underlying attributed graph model. Elision of the content of a container should result in the removal of hanging links and the display of the corresponding fused edges terminating on the container. In addition, the user should be provided with the option of making these fused attributes invisible to queries, so that for example the same edge is not shown terminating on a graph vertex and its parent container.

Although there is scope for the development of novel vertex layout strategies within the graph, attribute and blind categories identified in Section 3.2.2, much greater opportunities exist for the invention of hybrid techniques. Examples include: graph layout techniques which prioritise the removal of edge crossings or vertex occlusions on the basis of the attributes of the corresponding graph elements; attribute layouts which use synthetic edge attributes derived from graph-theoretic metrics such as path distance; and blind layouts which re-order vertices to optimally preserve some measure of vertex proximity.

The encapsulation of a layout composition tree within a macro which presents a user interface for only a subset of the layout strategy, selection expression and visual style parameters to the more time-constrained user would trade off customisation flexibility for the time required to fully specify a view. The addition of a shorthand mechanism for specifying recursive structures in the layout composition tree – such as that a directory contains files and directories – would also reduce the time required to specify a view.

When specifying a visual set, the user is currently provided with the ability to specify a query on the selected edge or vertex attribute. In future, the visually-specified, graph-based query mechanism provided for selection expressions should be made available for

the more flexible specification of visual sets. This query mechanism is currently being extended to permit more advanced queries involving sorting, aggregation and the creation of synthetic attributes. In future it will also need to support the Boolean combination of queries specified on partially overlapping subgraphs.

It is currently left to the user to choose without restriction the visual styles associated with each visual set from the set of all possible styles. Consequently, the styles applied to a vertex or edge are not necessarily perceptually "orthogonal", so that for example the choice of colour can affect the perception of size. Contradictory settings of the same visual attribute are also possible. The provision of palettes of visual attribute combinations, or agent-based support for the choice of visual attributes, could be used to overcome this problem.

6 Conclusion

In this paper we have presented the CLOVIS view family for visualising information which can be modelled as an attributed graph. A user interface for creating and interacting with CLOVIS views has been described, and the supporting InVision component infrastructure outlined. A framework for composing layouts has been presented, including the responsibilities of the layout strategy and the mechanism for coordinating the execution of layouts. The versatility of the CLOVIS view family was demonstrated in Section 4 through its application to a variety of problem domains, and future research directions identified in Section 5.

References

1. Herman, I., Melançon, G. & Marshall, M. S. (2000) Graph visualization and navigation in information visualization: A survey, *IEEE Transactions on Visualization & Computer Graphics* **6**(1), 24–43.

2. Card, S. K., Mackinlay, J. D. & Shneiderman, B., eds (1999) *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann Publishers, Inc., San Francisco, California.

3. Fairchild, K. M., Poltrock, S. E. & Furnas, G. W. (1988) SemNet: Three-dimensional graphic representations of large knowledge bases, *in Cognitive Science and its Applications for Human-Computer Interaction*, pp. 201–233.

4. Eick, S. G. & Wills, G. J. (1993) Navigating large networks with hierarchies, *in Proc. IEEE Visualization '93 Conf.*, pp. 204–210.

5. Kolojejchick, J., Roth, S. & Lucas, P. (1997) Information appliances and tools in Visage, *IEEE Computer Graphics and Applications.*

6. Olston, C., Woodruff, A., Aiken, A., Chu, M., Ercegovac, V., Lin, M., Spalding, M. & Stonebraker, M. (1998) Datasplash, *in Proceedings of the ACM SIGMOD International Conference on Management of Data*, Seattle, WA, pp. 550–552.

7. Olston, C., Stonebraker, M., Aiken, A. & Hellerstein, J. (1998) VIQING: Visual Interactive QueryING, *in Proceedings of the 14th IEEE Symposium on Visual Languages*, Halifax, Nova Scotia, Canada.

8. i2 (2000) Analyst's workstation, http://www.i2group.com/products/anw/index.htm.

9. Hartmann, S. (1995) Graph-theoretical methods to construct entity-relationship databases, *in* M. Nagl, ed., *Graph Theoretic Concepts in Computer Science: Proceedings of the Twenty-First International Workshop*, Vol. 1017 of *Lecture Notes in Computer Science*, Springer-Verlag, Aachen, Germany.

10. Cox, T. & Cox, M. (1994) *Multidimensional Scaling*, Chapman Hall, London.

11. Tufte, E. R. (1983) *The Visual Display of Quantitative Information*, Graphics Press.

12. Furnas, G. W. (1997) Effective view navigation, *in Proc. CHI'97, ACM Conference on Human Factors in Computing Systems*, Atlanta, Georgia, pp. 367–374.

13. Johnson, B. & Schneiderman, B. (1991) Tree-maps: A space-filling approach to the visualization of hierarchical information structures, *in Proc. IEEE Visualization '91 Conf.*, IEEE CS Press, pp. 275–282.

14. Geary, D. M. (1997) *Graphic Java 1.1: Mastering the AWT*, The Sunsoft Press Java Series, second edn, Sun Microsystems Press, chapter 14.

15. Bertault, F. (1995) Adocs: A drawing system for generic combinatorial structures, *in* F. Brandenburg, ed., *Symposium on Graph Drawing*, Vol. 1027 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 24–27.

16. Marshall, M., Herman, I. & Melançon, G. (2000) *An Object-Oriented Design for Graph Visualization*, Technical Report INS-R0001, Centre for Mathematics and Computer Sciences, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands. http://www.cwi.nl/InfoVisu/papers/INS-R0001.pdf.

17. Feng, Q. (1997) *Algorithms for Drawing Clustered Graphs*, PhD thesis, The University of Newcastle, Department of Computer Science & Software Engineering.

18. Kimelman, D., Leban, B., Roth, T. & Zernik, D. (1994) Reduction of complexity in dynamic graphs, *in Proceedings of the Symposium on Graph Drawing GD '93*, Springer-Verlag.

19. Huang, M. & Eades, P. (1998) A fully animated interactive system for clustering and navigation of huge graphs, *in* S. Whitesides, ed., *Graph Drawing 98*, Lecture Notes in Computer Science 1547, Springer-Verlag, pp. 374–383.

20. Kamada, T. (1989) *Visualizing Abstract Objects and Relations: A Constraint-based Approach*, Vol. 5 of *Series in Computer Science*, World Scientific.

21. Ahlberg, C. & Wistrand, E. (1995) IVEE: An information visualization & exploration environment, *in Proceedings of IEEE Symposium on Information Visualization.*

22. Friedrich, C. & Eades, P. (2000) The Marey graph animation tool, *in Proceedings of Graph Drawing 2000.*

23. Pattison, T., Vernik, R., Goodburn, D. & Phillips, M. (2001) *Rapid Assembly and Deployment of Domain Visualisation Solutions*, Technical Report DSTO-TR-1100, Defence Science & Technology Organisation.

24. Melamed, B. (1998) *Design and Implementation of an Attribute Manager for Conditional and Distributed Graph Transformation*, Master's thesis, Computer Science Department, Technical University of Berlin.

25. Object Management Group (1999) OMG Unified Modeling Language specification, http://www.omg.org/technology/uml/. version 3.

26. di Battista, G., Eades, P., Tamassia, R. & Tollis, I. (1999) *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall.

27. Melançon, G. & Herman, I. (1998) *Circular drawings of rooted trees*, Technical Report INS-R9817, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands.

28. Knuth, D. E. (1984) *The TEXbook*, Addison-Wesley, Reading, Mass.

29. Vanhelsuwé, L. (1997) *Mastering JavaBeans*, Sybex.

30. Derthick, M. & Roth, S. F. (2001) Enhancing data exploration with a branching history of user operations, *Knowledge Based Systems* **14**(1-2), 65–74.

31. Vernier, F. & Nigay, L. (2000) Modifiable treemaps containing variable-shaped units, *in* T. Ertl, B. Hamann & A. Varshney, eds, *CD-ROM Proceedings of the IEEE Visualization 2000 Conference*, IEEE, IEEE Press, Salt Lake City, Utah.

32. Vernik, R. & Burke, M. (1993) Perspective-oriented description: Integrating and tailoring information for software engineering, *in Proceedings of Sixth Conference on Software Engineering and its Applications*, Paris, France.

33. Vernik, R. J. (1996) *Visualisation and Description in Software Engineering*, PhD thesis, Dept. Computer and Information Science, University of South Australia, Adelaide, South Australia.

DISTRIBUTION LIST

Information Visualisation using Composable Layouts and Visual Sets

Tim Pattison, Rudi Vernik & Matthew Phillips

	Number of Copies
DEFENCE ORGANISATION	
Task Sponsor	
Director General C4	Doc Data Sht
Deputy Director, Battlespace Digitisation	1
S&T Program	
Chief Defence Scientist FAS Science Policy AS Science Corporate Management Director General Science Policy Development	1
Counsellor, Defence Science, London	Doc Data Sht
Counsellor, Defence Science, Washington	Doc Data Sht
Scientific Adviser to MRDC, Thailand	Doc Data Sht
Scientific Adviser Joint	1
Navy Scientific Adviser	Doc Data Sht & Dist List
Scientific Adviser, Army	Doc Data Sht & Dist List
Air Force Scientific Adviser	1
Director Trials	1
Aeronautical and Maritime Research Laboratory	
Director, Aeronautical and Maritime Research Laboratory	1
Electronics and Surveillance Research Laboratory	
Director	Doc Data Sht & Dist List
Chief, Information Technology Division	Doc Data Sht
Research Leader, Military Information Enterprise Branch	1
Research Leader, Advanced Computer Capabilities Branch	Doc Data Sht
Research Leader, Command & Control and Intelligence Systems Branch	Doc Data Sht
Research Leader, Joint Systems	1
Head, Information Warfare Studies Group	Doc Data Sht
Head, Enterprise Visualisation, Instrumentation & Synchronisation Group	1
Head, Trusted Computer Systems Group	Doc Data Sht

Head, Systems Simulation and Assessment Group	Doc Data Sht
Head, C3I Operational Analysis Group	Doc Data Sht
Head, Information Exploitation Group	Doc Data Sht
Head, Intelligence Group	Doc Data Sht
Head, Human Systems Integration Group	Doc Data Sht
Head, C2 Australian Theatre Group	1
Head, Distributed Systems Group	Doc Data Sht
Head, C3 Information Systems Concepts Group	1
Head, Military Systems Synthesis Group	Doc Data Sht
Head, Systems of Systems Group	Doc Data Sht
Head, Operational Information Security Group	Doc Data Sht
Head, Advanced Network Integrity Group	1
Publications & Publicity Officer, ITD	1
Executive Officer, ITD	
Author	10
DSTO Research Library and Archives	
Library Fishermans Bend	Doc Data Sht
Library Maribyrnong	Doc Data Sht
Library Salisbury	1
Australian Archives	1
Library, MOD, Pyrmont	Doc Data Sht
US Defense Technical Information Center	2
UK Defence Research Information Centre	2
Canada Defence Scientific Information Service	1
NZ Defence Information Centre	1
National Library of Australia	1
Capability Systems Staff	
Director General Maritime Development	Doc Data Sht
Director General Aerospace Development	Doc Data Sht
Knowledge Staff	
Director General Command, Control, Communications and Computers (DGC4)	Doc Data Sht
Director General Intelligence, Surveillance, Reconnaissance and Electronic Warfare (DGISREW) R1-3-A142 Canberra ACT 2600	Doc Data Sht
Director General Defence Knowledge Improvement Team (DGDKNIT) R1-5-A165, Canberra ACT 2600	Doc Data Sht

Army

Stuart Schnaars, ABCA Standardisation Officer Tobruk Barracks, Puckapunyal VIC 3662	4
SO (Science), Deployable Joint Force Headquarters (DJFHQ)(L)	Doc Data Sht

Intelligence Program

DGSTA, Defence Intelligence Organisation	1
Manager, Information Centre, Defence Intelligence Organisation	1

Corporate Support Program

Library-Manager, DLS-Canberra	1

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy Library	1
Head of Aerospace and Mechanical Engineering, ADFA	1
Deakin University Library, Serials Section (M List)	1
Hargrave Library, Monash University	Doc Data Sht
Librarian, Flinders University	1

OTHER ORGANISATIONS

NASA (Canberra)	1
AusInfo	1
State Library of South Australia	1
Parliamentary Library of South Australia	1

ABSTRACTING AND INFORMATION ORGANISATIONS

Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, US	1
Materials Information, Cambridge Scientific Abstracts, US	1
Documents Librarian, The Center for Research Libraries, US	1

INFORMATION EXCHANGE AGREEMENT PARTNERS

Acquisitions Unit, Science Reference and Information Service, UK	1
Library – Exchange Desk, National Institute of Standards and Technology, US	1

SPARES

DSTO Salisbury Research Library	5

Total number of copies: 58

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA				1. CAVEAT/PRIVACY MARKING
2. TITLE Information Visualisation using Composable Layouts and Visual Sets			3. SECURITY CLASSIFICATION Document (U) Title (U) Abstract (U)	
4. AUTHORS Tim Pattison, Rudi Vernik & Matthew Phillips			5. CORPORATE AUTHOR Electronics and Surveillance Research Laboratory PO Box 1500 Salisbury, South Australia, Australia 5108	
6a. DSTO NUMBER DSTO–RR–0216	6b. AR NUMBER AR-011-960		6c. TYPE OF REPORT Research Report	7. DOCUMENT DATE August, 2001
8. FILE NUMBER N9505/21/42	9. TASK NUMBER JNT 00/130	10. SPONSOR DGC4	11. No OF PAGES 24	12. No OF REFS 33
13. URL OF ELECTRONIC VERSION http://www.dsto.defence.gov.au/corporate/reports/DSTO–RR–0216.pdf			14. RELEASE AUTHORITY Chief, Information Technology Division	

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved For Public Release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, SALISBURY, SOUTH AUSTRALIA 5108

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS

No Limitations

18. DEFTEST DESCRIPTORS

Situation awareness
Software architecture
Visualisation
Complex systems
Software tools

19. ABSTRACT

A modern military enterprise is characterised not only by its people, physical infrastructure and geography, but also by its business processes, knowledge management practices and fluid organisational structures. Management, coordination, planning and development of the enterprise all require awareness of its current state. To aid these functions, the DSTO task JNT 00/130 entitled "Assembly and Deployment of Defence Visualisation Solutions" (ADDVIS) proposes the use of information visualisation techniques to produce integrated enterprise situation awareness pictures tailored to meet the requirements of ADF functions such as capability development, system management and self-synchronisation. To this end, Information Technology Division (ITD) is performing research and development (R&D) into the next generation of information visualisation systems which will enable the rapid assembly and deployment of Defence visualisation solutions. InVision is a component-based software architecture for the rapid prototyping of information visualisation solutions. Its evolutionary implementation has given rise to an experimental component infrastructure with the aid of which the assumptions and goals of the ADDVIS task are being tested. This report describes the CLOVIS class of views, along with the associated supporting InVision infrastructure. The versatility and generality of the CLOVIS class of views described in this report makes it ideal for rapid prototyping of information visualisations. Its subsumption of a number of existing information layouts constitutes significant progress towards one of the key goals of InVision: the integration of various visual representations, with each chosen on the basis of its particular suitability to the task at hand.

RESEARCH REPORT DSTO-RR-0216 AR-011-960 AUGUST 2001

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE & TECHNOLOGY ORGANISATION DSTO

ELECTRONICS AND SURVEILLANCE RESEARCH LABORATORY
PO BOX 1500 SALISBURY SOUTH AUSTRALIA, 5108
AUSTRALIA, TELEPHONE (08) 8259 5555