Visual Representation of Complex Information Structures in High Volume Manufacturing

Connor Upton and Gavin Doherty

Distributed Systems Group, Department of Computer Science, Trinity College Dublin
{connor.upton, gavin.doherty}@cs.tcd.ie

Abstract. While research supports the use of graphic data representations in interfaces and control systems, work in this area has focused on relatively small systems with a limited number of variables. This paper describes an approach to designing a visual application for a semiconductor manufacturing plant. This is a complex, large-scale system requiring a structured design methodology. First, using cognitive work analysis techniques an Abstraction Decomposition Space (ADS) of the system is generated. Second, as with ecological interface design, we demonstrate how this ADS can inform the display design. The complexity and scale of the system has required us to make adjustments to both of these frameworks. The resulting display req multiple views of the system, information hiding and user interaction. Tak wider set of analyses onboard, we present a design rationale supportin explicit representation of hierarchies, the compatibility of views and the u contextual navigation.

1 Introduction

The visual representation of data in control interfaces has been shown to increase performance and reduce human error [1, 2]. Research into the area of external cognition has indicated that the correct graphical encoding of data allows us to exchange cognitive operations for perceptual operations thus improving performance [3]. Techniques for the encoding of data using visual variables have been explained and validated [4, 5]. While the knowledge gained from this work can be used to inform visual design, much of the experimental work has focused on small problem spaces involving a limited number of variables [3, 6]. In these situations improved performance can be attributed to the externalisation of low-level cognitive operations such as search, comparison and integration, freeing up short term memory to focus on higher level tasks.

For interactive systems a structured approach to design is required. Data representations should not be created in isolation but must be considered in relation to the overall system. Various different approaches to user studies, requirements gathering and design evaluation already exist. While these are helpful in defining information requirements and useful for post-hoc design evaluations, there is relatively little information on how to transform information requirements into visual displays. This has been referred to as "the Design Gap" [7]. The design gap can be attributed to the large number of factors that effect visual displays and the diverse areas of research which are relevant to this field. In this paper we present a complex

Please use the following format when citing this chapter:

Upton, C., Doherty, G., 2006, in IFIP International Federation for Information Processing, Volume 221, Human Work Interaction Design: Designing for Human Work, ed. T. Clemmensen, Campos, P., Orngreen, R., Petjersen, Al., and Wong, W., (Boston: Springer), pp. 27–45.

socio-technical system and outline an approach for revealing user requirements. We then present a design rationale that creates a visual display informed by our initial system analysis coupled with a range of design guidelines, thus bridging the design gap for this particular problem.

1.1 Complex Sociotechnical Systems

Complex sociotechnical systems are work environments that involve large problem spaces, multiple users & high-levels of automation. As open systems they frequently feature conflicting constraints, dynamic data, coupled components and unanticipated events. Examples of such environments include industrial process control, air traffic control and surgery theatres. Workers in these environments observe and interact with large volumes of real-time system data. This data is often multivariate with complex relationships existing between data sets. The cognitive activities involved in such systems go far beyond the low-level operations mentioned above. The complexity and scale of these systems means that data must be structured in a manner that is meaningful to end users before it can be visualised.

1.2 Design Framework

Ecological Interface Design (EID) [8] is a framework for designing interfaces for complex systems. It takes a two step approach first specifying the content to be displayed and then designing the visual form. While the framework is well defined it can be extended in a number of ways.

Firstly, the framework was originally defined in relation to process control. In order to test the frameworks generalisability it is important to apply it to different domains. Here we apply it to a High Volume Manufacturing environment.

Secondly, the example in the original framework deals with a representational microworld. A microworld is a pared down version of a real world system. It maintains aspects of a systems complexity while being simple enough to carry out accurate controlled experiments. The DuressII [8, 9, 10] system, around which the EID framework was developed, benefited from having a relatively small set of variables, a simple physical structure and a single operator. Our study deals with the large scale, highly complex process flow involved in semiconductor manufacturing.

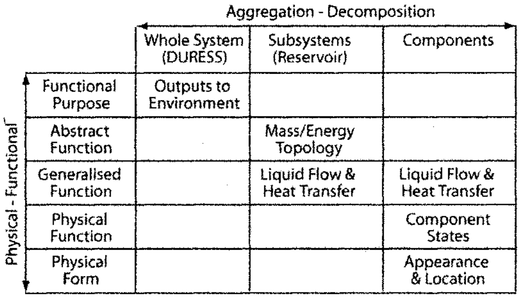
Thirdly, while the principles of visual design proposed by the framework provide us with guidelines, they do not deal with the actual visual rendering of the components. This allows the framework to remain general. With smaller domains the principles can be achieved through rudimentary graphic techniques such as spatial arrangement. However large scale systems require more sophisticated representational techniques. We discuss the application of these techniques to the High Volume Manufacturing domain.

2 The EID Framework

Here the two stages involved in the EID framework are explained. The first stage involves the specification of content through the generation of an Abstraction Decomposition Space (ADS) [9] a tool used in cognitive work analysis. The second stage applies three principles of visual design to the content in the ADS to inform the interface design.

2.1 Specification of Content

Cognitive Work Analysis (CWA) [10] is a methodology created to analyse complex socio-technical systems. It takes a different analytical approach to other Human Computer Interaction research methods. Instead of looking at a work domain in terms of users and specific contexts of use; it aims to describe the complete domain in terms of the systems constraints. This generates a field description of the entire system. A good analogy for explaining field descriptions is that of helping someone to find their way to a location. User focused research will describe how a task is completed by an individual. It specifies a number of actions carried out to complete the task. It can be compared to giving someone a set of verbal instructions on how to get to a location, take the first right, then the second left etc. A field description describes the constraints under a task can be completed. It is more comparable to handing someone a map with their current location and target marked. This allows them to find their way but also to adjust their route should the need arise. A field description accommodates different worker roles in the same system. It can deal with non-normative work scenarios and can provide a more accurate model of the system.


Aggregation - Decomposition
Physical - Functional

	Whole System (DURESS)	Subsystems (Reservoir)	Components
Functional Purpose	Outputs to Environment		
Abstract Function		Mass/Energy Topology	
Generalised Function		Liquid Flow & Heat Transfer	Liquid Flow & Heat Transfer
Physical Function			Component States
Physical Form			Appearance & Location

Fig. 1. The Abstraction Decomposition Space for DuressII [10]

The Abstraction Decomposition Space (ADS) [11, 12] is an analytical tool used in CWA to create a field description of the work domain by combining a decomposition hierarchy with an abstraction hierarchy. A decomposition or part-whole hierarchy splits a system into its subsystems and then subsystems into components. It reduces complexity by dividing a system into smaller units. An abstraction hierarchy is a description of a system in terms of functionality, from high-level goals down to the physical description of individual components that carry out basic physical tasks. Rasmussen proposes five divisions for process control; functional purpose, abstract function, general function, physical function and physical form. The ADS places these hierarchies orthogonally against each other providing us with a multilevel view of the system where each level describes the entire system at a different granularity. DuressII was a thermal-hydraulic process simulation around which the EID framework was developed. Figure 1 shows the ADS for this system. The ADS can be used to define the information requirements of individual users through use-case mappings [10].

2.2 Design of Visual Form

The EID framework outlines three principles for visual design of interfaces based on the Skills, Rules, Knowledge (SRK) taxonomy [13]. This taxonomy defines three levels of cognitive control that a user can exert over a system, distinguished by the manner in which the system is represented internally. Skills Based Behavior (SBB) involves reactive behavior to real-time system data. It describes how an expert user responds to temporal information about system components to maintain stability. Generally these actions are so fluid that they become instinctive and are not verbalised by users. Rule Based Behavior (RBB) relates to procedural tasks that follow a plan of action. They are guided by rules defined by the constraints of the system. While they can be learned and practiced to the point of fluency the actions can generally be recognised and verbalised by the user. Knowledge Based Behavior (KBB) relates to higher-level decision making. It generally involves reasoning at multiple levels of abstraction and requires knowledge of the complete relational structures in the system. KBB is used in fault diagnosis and performance analysis.

2.2.1 Levels of Cognitive Control

The way in which information is interpreted facilitates different levels of cognitive control. Vicente provides a good example with the controlling of a valve using a flow-meter [10]. The flow-meter consists of a measurement scale, a target indicator and a pointer. Given the task of stabilising the flow at the target the flow-meter becomes a signal. The time and space data given through the flow-meter gives the operator temporal feedback facilitating control. This is Skills Based Behavior. If the user closes the valve but the meter still indicates a flow, it becomes a sign. The mapping between the valve and the feedback from the flow-meter indicates that there is a mis-calibration between the meter and the valve. This is Rule Based Behavior. If the user recalibrates and the problem arises again the meter becomes a symbol. It becomes an element in the overall system that is related to other components and through it we can diagnose a leak in the valve or another fault in the system. This is an example of Knowledge Based Behavior.

2.2.2 Principles of Visual Design

The flow-meter example describes cognitive control of a single component through a physical interface. With modern sensor and display technology it is possible to integrate the control and feedback display for entire systems into a single digital interface. The EID framework notes that careful consideration must be given to the representation of system data to ensure safety and efficiency of use. Three principles of visual design are provided, each associated with supporting a level of cognitive control.

1. SBB – to support temporal control of a system direct manipulation should be used. Also the representation should be isomorphic to the part-whole structure.
2. RBB – provide a consistent one-to-one mapping between constraints and the cues or signs provided by the interface.
3. KBB – represent the work domain in the form of an abstraction hierarchy to serve as an externalised system model.

2.3 Application of EID

As a representative microworld DuressII sought to capture many of the characteristics of process-control in complex systems, while remaining simple enough to facilitate experiments. To achieve this, the complexity of the system was limited. The presence of a dual supply system introduced the complexity associated with coupled components. Despite this coupling, the overall process can still be identified as having a linear supply & demand relationship. The scale of the system was also limited. A small number of subsystems and components gave a total of thirty seven system variables. Finally, the system was designed for a single operator.

The limited scale and complexity of the system made it easier to follow the design principles set out by EID. SBB was satisfied by providing visual representations of the component related data and their associated controls on-screen. RBB was satisfied by showing all components in a single view and by making their coupling visually explicit. This would not have been possible with a larger system or more complex coupling. KBB was supported by grouping components according to their subsystem. This allowed the ADS to be represented by way of visual chunking.

In the remainder of this paper we apply the EID approach to the domain of High Volume Manufacturing. We note a number of differences between the two systems and make adjustments to the EID approach in order to accommodate them.

3 Case Study: Semiconductor Manufacturing

Modern High Volume Manufacturing (HVM) environments are examples of extremely complex socio-technical systems. They combine sophisticated factory automation with the changing demands of dynamic markets. A common constraint across HVM is the conflicting goals of achieving high volumes of production while ensuring that equipment continues to operate within acceptable quality control limits. High production volumes place equipment under stress, which may affect the quality of the product and the overall product yield. This requires equipment to receive more maintenance and repair. However, repair causes more downtime leading to lower levels of production. This conflict is generally resolved by humans who must reconcile manufacturing (production) focused and engineering (quality) focused priorities.

3.1 Description of System

Semiconductor Fabrication Plants (Fabs) are HVM environments involving hundreds of machines (described as tools) and a highly complex process-flow. The overall production process between raw silicon and final semiconductor is divided into a number of segments. Segments can be further subdivided into a number of functional operations that build parts of the semiconductor device. As semiconductor manufacturing is a multilayered process, some operations may be repeated in different segments, introducing circulation and re-entries into the process-flow. Operations are carried out on specific tools which are categorised according to specific functional activities; for example etching or lithography. Multiple tools carrying out the same operation are gathered together into a toolset. Groups of toolsets that carry out the same general function form a functional area. This complex relationship between process-flow and functional areas is shown in fig.2.

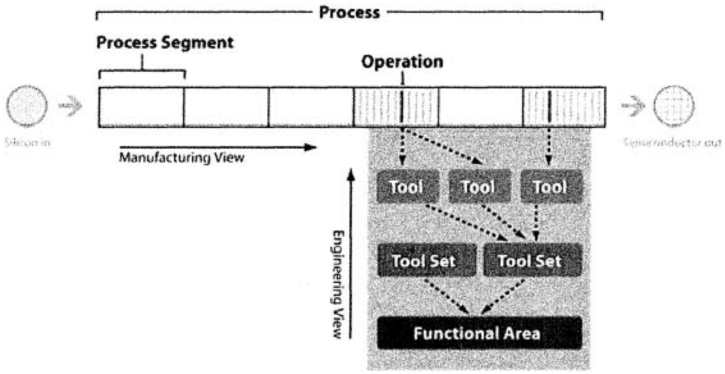


Fig. 2. Relationship between Process-Flow and Functional Area

Process flow is a manufacturing concept while functional areas are related to engineering concerns. This relationship indicates how the conflicting goals are disseminated throughout the system.

3.2 Social Organisation

Responsibility for controlling the Fab is spread across the social organisation of workers. Two of the main structures within the social organisation are manufacturing and engineering, mirroring the conflicting goals mentioned above. Within these structures a management hierarchy exists. Factory floor workers focus on smaller parts of the overall system and have limited information requirements for carrying out their jobs. Management level workers make decisions that relate to larger portions of the overall system. These decisions are directly affected by the production/quality conflict.

3.3 Development of Whole System Interface

Currently the system is operated through a range of individual applications. These have been developed to support specialist operations and provide only partial views of the overall system. Control applications exist at the tool level providing data relating to tool performance. While this allows an operator to control their tools, it is difficult to relate the data back to higher level metrics that inform us about the overall system state. The result of this set-up is that information must be combined from multiple sources to gain an understanding of system performance at higher levels of abstraction. Currently this is done through the manual generation of reports and verbal communication at management meetings.

The current information systems support the temporal control of system components. We can think if this as enterprise level SBB. However, enterprise level KBB cannot be supported through these systems as there is no interface that explicitly represents the relational structures of the fab. Such an interface would have a number of benefits over the current reporting tools. It would make it easier to see the effect of the conflicting goals at higher levels of abstraction, it would enable users to quickly drill-down and locate the cause of these conflicts and it could allow users to recognise patterns in system behavior. Here we attempt to use the EID framework to generate such an interface.

4 Development of ADS for HVM

We have discussed how the ADS for the DuressII system was generated by combining a functional abstraction hierarchy with a physical decomposition. In the Fab environment the physical decomposition has limited use. While physical tools match functional operations at the lowest levels, circulation in the process-flow means that physical and functional relationships no longer equate at higher levels of abstraction. In figure 2 we can observe that values associated with a process segment have no relation to values associated with a toolset. If a physical decomposition is no longer valid what other constraints can we use? The decomposition hierarchy is not limited to physical constraints. Other examples of system constraints that may be used include functional purpose, organisational and legal constraints [8].

4.1 Two Decomposition Hierarchies

We have already outlined two functional constraints of the system that are common across HVM environments. Both of these have had a direct effect on the organisational structure and both can provide decomposition hierarchies. The manufacturing hierarchy organises the system into different levels of granularity according to product position in the process-flow. The hierarchy consists of a process which is divided into segments which in turn are subdivided into operations. This facilitates a horizontal view across the process-flow. The engineering hierarchy allows us to think about the system in terms of equipment. It provides us with a vertical view down into functional areas, toolsets and tools (fig. 2).

4.2 The Abstraction Lattice

Both of the views mentioned above are valid system decompositions and can be used to generate independent ADSs of the system. However, as conflicting goals they are non-analogous. Items in similar positions in their individual ADS structures may not be related. How can we integrate these into a single model of the system that displays both structures? While they are very different at the abstraction level of functional purpose, they share the same properties at the level of physical form. This commonality can act as a bridging point between the two views. This allows us to develop our model as an Abstraction Lattice (fig.3). An Abstraction Lattice allows us to reason our way down through levels of abstraction in one view and then up through levels of abstraction in an alternative view of the same system. This approach allows us to reflect the Abstraction Hierarchy across the level of physical form joining up the two ADS representations. Our new ADS (fig.4) captures all of the system variables from both view at multiple levels of abstraction. This completes the first stage of the EID process. This ADS gives us a field description of the system and can be used to carry out use-case scenario mappings for various tasks [14].

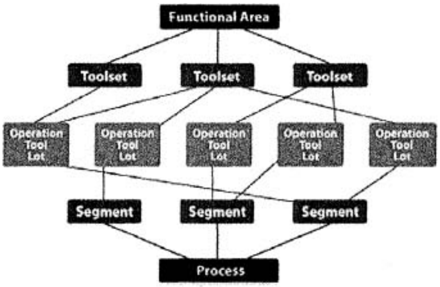
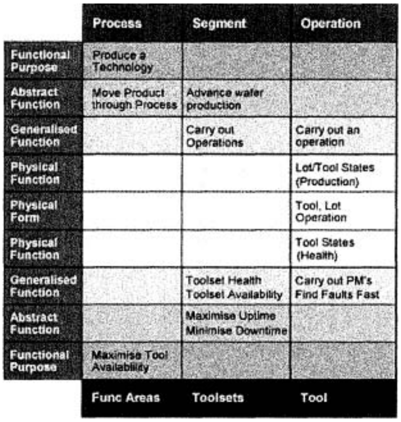


Fig. 3. The Abstraction Lattice



	Process	Segment	Operation
Functional Purpose	Produce a Technology		
Abstract Function	Move Product through Process	Advance wafer production	
Generalised Function		Carry out Operations	Carry out an operation
Physical Function			Lot/Tool States (Production)
Physical Form			Tool, Lot Operation
Physical Function			Tool States (Health)
Generalised Function		Toolset Health Toolset Availability	Carry out PM's Find Faults Fast
Abstract Function		Maximise Uptime Minimise Downtime	
Functional Purpose	Maximise Tool Availability		
	Func Areas	Toolsets	Tool

Fig. 4 . ADS for Fab

4.3 Information Requirements

Our goal is to facilitate KBB in relation to the overall system. To allow for this we need to supply an interface that can achieve a number of goals. Firstly, it must communicate information relating to both views of the system. Secondly, it must represent information from these views at different levels of abstraction. Thirdly, it must allow the user to combine information across the views at different levels of abstraction. In the next section we show how this modified ADS requires a different approach to visual representation than the approach used with the DuressII system.

5 Design of Visual Form

As previously discussed, the DuressII system used data chunking to embed the abstraction hierarchy into a visual design. A number of factors made this possible. Firstly, the ADS of the system used a physical decomposition hierarchy. In the interface subsystems were indicated through the physical clustering of components and traversal of the ADS was made possible through visually focusing on specific areas of the overall display. Secondly, the relatively small scale of the system meant that it was physically possible to display all of the variables in a single screen.

5.1 Differences between Microworld & Real World Applications

The Fab is a much more complex system. As we have seen, circulation in the process-flow means that a physical decomposition does not provide us with a useful model of system functionality. Our ADS uses two decomposition hierarchies based on the conflicting functional constraints of the system. While these hierarchies are related at certain levels they are not analogous and cannot be combined into a single graphic form, therefore both must be embedded in the display if KBB is to be supported. The scale is also different. While microworlds deal with a small number of variables, the fab features thousands of variables of different data types. Limitations of perception and cognition make it difficult for humans to work with this number of variables of mixed data types and complex structures. Bertin's "impassible barrier" indicates how it is impractical to represent relationships in data with more than three variables in a single image [4]. Furthermore, if we are to allow for direct manipulation of these variables, there is a lower-limit to the physical size we can represent them. This limits the number of variables that can be represented on a single screen. In order to represent the data we need to hide information within the levels of abstraction supplied in the ADS. Higher-level metrics or summary data can act as gateways into specific lower-level data. A challenge that arises with this is how to avoid "keyholing" [1], where the presentation of detailed information hides its context in an overall system.

5.2 Properties of Final Display

The differences in scale and complexity mean that visual chunking cannot be used as a technique for embedding our ADS in our final interface. This is further supported by the fact that our views are non–analogous, meaning that both hierarchies must be represented. We have defined three goals that our final interface must achieve if it is to support KBB.

1. Both abstraction hierarchies must be displayed
2. The hierarchies must be differentiable and compatible
3. Cross-hierarchy relationships must be made explicit

6 Representation of Hierarchies

Hierarchies are organisational systems that allow us to think about data at higher levels of abstraction. Different types of hierarchies include taxonomies, organisational structures and file systems. Tree structures, consisting of nodes, connections and leaves are the most basic way to visualise hierarchies. Nodes are organisational structures that can contain other nodes or leaves. Connections indicate the relationship between nodes. Leaves are low-level data that cannot be subdivided. Two core concepts have been used to define the graphical representation of hierarchies', connection and enclosure. [15]

6.1 Visual Hierarchies

Connection (fig. 5) uses the most literal visual representation of the tree structure. Here nodes are linked to sub-cases by lines indicating the connections. The structure is encoded on the spatial axes with the x axis carrying the nominal variable (N) and the y axis indicating the ordinal variable (O) of hierarchical level. The use of the spatial axes means that the representation can get quite unwieldy when dealing with large hierarchies.

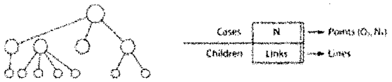


Fig. 5 . Connection Tree Structure [15]

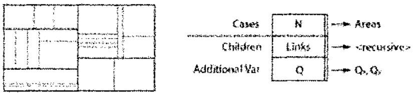


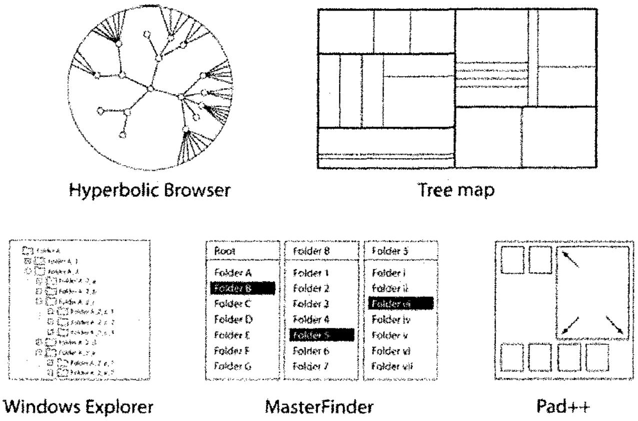
Fig. 6. Enclosure Tree Structure [15]

Enclosure uses area to represent nodes. Child nodes are contained within parent nodes in a recursive manner. Thus connections are indicated through enclosure. This approach allows us to use the area of nodes to convey quantitative data (Q) associated with the system. Again this techniques runs into difficulty when displaying large hierarchies as the deepest cases become very small and difficult to see.

The visual chunking of components in the DuressII system is a form of the enclosure technique. The need to represent quantitative component variables makes the connection technique unsuitable. Given the existence of two hierarchies we propose a dual display approach for our interface allowing us to divide up these representative tasks. This allows us to think about the visual representation of each hierarchy independently.

6.2 Suitable Representation is Defined by Task

The concepts above deal with visually representing the structure of a hierarchy. Having a view of the overall structure of a hierarchy is important for developing a full understanding of the system it represents, however this is only one use of it uses. The visual representation of hierarchies and their contents facilitates a number of tasks including: perceiving balance and connectivity of structures; making comparisons and navigating between levels; transferring content between levels; making comparisons and navigating within a level and; transferring content within a level. While many different techniques for displaying hierarchical information have been suggested (fig.7) they tend to support different tasks and actions to varying degrees of success. Here we review a set of common techniques and indicate the tasks they can support.



Hyperbolic Browser

Tree map

Windows Explorer

MasterFinder

Pad++

Fig. 7 . Hierarchy representation techniques

Hyperbolic Browser

Presentation technique: Graphical Tree
Nodes: Boxes, Connectors: Lines, Data: Not Represented
Tasks Supported: navigation, perceiving balance and connectivity
Restrictions: Shows overall structure but cannot carry additional metrics. Data must appear in a different window.

Tree map

Presentation technique: Nested boxes
Nodes: Boxes, Connectors: enclosed areas, Data: Area
Tasks Supported: Comparison within and across levels.
Restrictions: Recursive nature of display makes it difficult to see low level data.

Windows Explorer

Presentation technique: Semi-graphic Tree
Nodes: Folders, Connectors: Lines, Data: Not Represented
Tasks Supported: Browsing, Content Transfer
Restrictions: Only reveals a path in the overall structure. Use of labels makes it difficult to display the total structure. Data must appear in a different window.

MasterFinder

Presentation technique: Hierarchical Browser

Nodes: Window, Connectors: Adjacent window, Data: Filenames
Tasks Supported: Navigation, Content transfer between levels
Restrictions: Only shows path and related nodes through a hierarchy.

Pad++
Presentation technique: Zoomable User Interface
Nodes: Boxes, Connectors: enclosed areas, Data: text or graphic
Tasks Supported: navigation, Comparison within and across levels.
Restrictions: Zooming makes comparison of low level data difficult.

6.3 Analysing our Case Study Hierarchies

No single representation supports all possible tasks related to hierarchies with maximum efficiency. In order to understand what representational technique is best suited to our problem we analyse the hierarchies involved in our ADS. We examine each view under the following three terms: purpose and task, data involved and depth of hierarchy. Purpose and task can inform the general technique we use for representing the hierarchical structure. Knowledge of data types can be combined with rules of data representation [4, 5] to further inform the display. The depth of the hierarchy can indicate whether a list-style presentation of the hierarchy is possible.

6.3.1 Process view

Purpose & Tasks: This manufacturing based view is a horizontal view across the process flow. The purpose of the view is to see the volume of product in the plant. This should allow the user to see the productivity level of toolsets, segments and the overall fab. The view at the lowest level allows the user to spot potential bottle-necks in the process-flow. The tasks involve viewing product levels at an operation (toolset), comparing product levels between toolsets, viewing product distribution across the process, viewing product capacity levels in segments and viewing product capacity across the entire fab.

Data Involved: The structure of the hierarchy is based on the division and subdivision of the process flow into more manageable units namely process, segment and operation. As a part-whole hierarchy it is possible to think of operations as having a quantitative relationship to both its parent segment and the overall system. As the process flow is a series of consecutive steps the nodes within each level can be said to have an ordinal relationship with each other. The additional data variable to be displayed is volume of product. This is quantitative data.

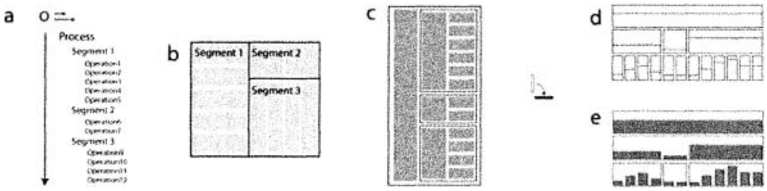


Fig. 8. Display design of process view

Depth of Hierarchy: The hierarchy is shallow consisting of only 3 levels process, segment and operation.

Design Rationale: With only three levels of depth the process hierarchy is relatively shallow and may be easily flattened. This allows us to convert it into a list representation making it easier to understand [16]. The ordinal relationship between nodes at all levels further supports this list representation as the use of a spatial axis supports ordinal data (fig. 8a). However, a list style presentation is not optimal for displaying the quantitative values associated with product volumes. Having segments interspersed with operations makes it difficult to carry out quantitative comparisons at these different levels of abstraction. An alternative approach is to use a treemap (fig. 8b). Here the higher level metrics are created naturally through the use of enclosure. Unfortunately the display then loses the ordinal relationship between the nodes necessary for comparison. A solution is a hybrid between the two. The enclosure technique is combined with a list to give a display that allows for both comparison at a level and between levels of detail (fig. 8c). The volume metric can then be expressed through position of point or area as both of these visual variables support quantitative information [4].

6.3.2 Functional View

Purpose & Tasks: The second display, the engineering based view, is a vertical view down into the system. The purpose of the view is to analyse the performance and availability of equipment across the fab. This involves looking at data from individual tools and toolsets and understanding the relationship between lower level data and higher level metrics. The tasks include accessing tool data, comparing tool data, controlling tool activity, transferring tools between toolsets, accessing and comparing toolset metrics.

Data Involved: The structure is a taxonomic hierarchy based on functional activity. This means that the relationship between nodes both vertically (down through the levels) and horizontally (across a level) is nominal. While different functional areas will have different numbers of toolsets, tools may operate at different speeds (throughput rates) so proportional or quantitative relationships between toolsets and functional areas are not very relevant. While the process view only needed to carry one additional variable (volume), functional view nodes have a number of associated variables relating to performance and availability at all levels. Some of these variables can be calculated into higher-level metrics relating to parent nodes. For example the health of a toolset is derived from the health of its associated tools which in turn derive their health from a set of sensor readings.

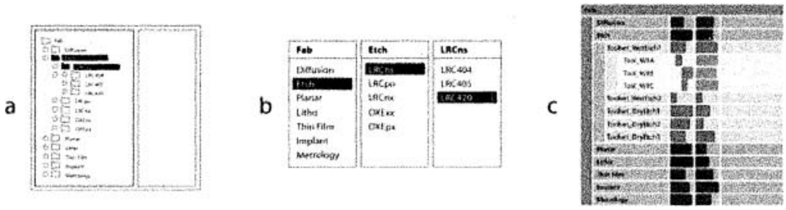


Fig. 9. Display design of functional view

Depth of Hierarchy: The functional hierarchy has two further levels not represented on our ADS. At the top level we have the overall Fab and at the lowest level we have individual tool variables that give us the tool health figures. This is a deeper structure consisting of five levels fab, functional area, toolset, tool and tool variables.

Design Rationale: As the structure is a deep taxonomical hierarchy it should not be flattened into a list form. The structural organisation is nominal not ordinal so a treemap type approach is possible. A treemap approach could also support a health metric through area and an availability-metric through colour. However, one of the tasks is the comparison of attributes at different levels. The arbitrary shapes generated by the treemaps space-filling recursive algorithm, coupled by the small display sizes at the lowest levels of granularity makes this comparison difficult. What we require is a path through the hierarchy displaying metrics at each level of granularity. The windows explorer (fig. 9a) allows us to unfold hierarchies through interaction, but it only displays data for one active node at a time and this appears in an adjacent window. The hierarchical browser (fig. 9b) also unfolds a hierarchy but it achieves it by opening a new adjacent list of nodes each time a node is accessed. Our display needs to combine the information hiding techniques of the windows explorer with the display capabilities of the hierarchical browser. However rather than displaying all of the associate nodes in the path we display only the metrics for the nodes associated with the path. By aligning these metrics along a horizontal axis it should become possible to read and compare performance across levels in the hierarchy as well as drilling down into the hierarchy to reveal the explanation for the metrics (fig. 9c).

6.4 Making Views Compatible

Achieving Knowledge Based Behavior requires information from both views of the system and at various levels of abstraction. We identify three qualities that our representations must have in order to achieve this. The representations must allow the user to:

1. Easily differentiate between the two hierarchies
2. Relate low-level data to higher level metrics
3. Identify relationships between views

6.4.1 Visually Differentiate between Hierarchies

The abstraction hierarchies for both views are similar in form and scale. This makes it possible to carry out use-case mappings and reveal information requirements [14]. However, the similarities must end here. The two views relate to different information and are used in different ways. Workers in manufacturing have a greater interest in process flow while engineering is more concerned with the system health. While many workers will benefit from having both views, it is critical that their visual representations make them easily discernible. There are a number of reasons for this.

Firstly, similar representations would require labeling and the reading of labels to allow for differentiation. This increases the cognitive work load of the user and would reduce performance. Secondly, a similar rendering of the hierarchies would give a false impression of correlations between levels of abstraction. This would give a false interpretation of the data relationships as the hierarchies are non-analogous and this correlation does not exist. Thirdly, a resemblance between the views could lead the viewer to access the wrong information. An example of this would be browsing through a process view when looking for health information.

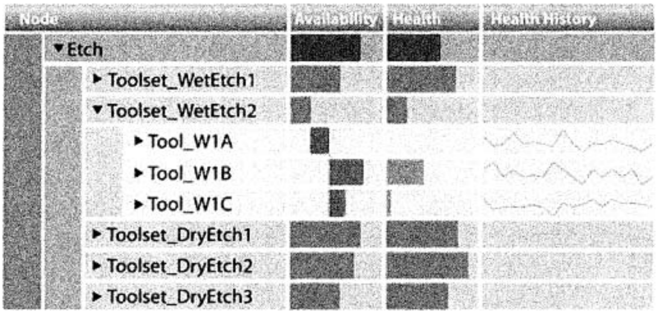
We have seen that there are many different ways to visually represent hierarchies. It is important that we select methods that are visually different to avoid the problems listed above. In our case the tasks associated with the hierarchies and the data involved have already guided us toward different visual representations.

6.4.2 Relate Low-level Data to High-level Metrics

Abstraction hierarchies allow us to structure complex systems and help in the development of conceptual models. They also enable us to abstract low-level data into higher level metrics allowing us to view system state information at different levels of detail. It is important that the visual representation allows us to understand the hierarchies involved. Different display techniques can achieve this in different ways.

In the proposed process view (fig. 8) we need to represent the product volumes at the level of operation, segment and overall process. Volume is a quantitative variable and can be represented through either spatial position (fig. 8d) or area (fig 8e). While the spatial position of a point is a valid representational format, its small surface area makes it difficult to see volumes across multiple areas. The other alternative, using area, allows us to see waves of product volumes across the process. It allows us to make visual estimations of volumes between segments and can also be used at the level of segment and process to indicate product levels with regard to a maximum volume.

The functional view is somewhat different. Multiple variables exist at each level of abstraction. These relate to a number of different metrics including availability (time to next scheduled maintenance), health and health history (fig. 10). All of this information is important for diagnostic tasks allowing a user to search and compare data in order to find causes of faults. Readings at any specific level may be horizontal or vertical. For example a user with responsibility for the etch area may look across the etch data and see that the health level is lower than expected. By clicking on the etch label the user can expand the matrix to reveal the lower level toolset data. Now, by scanning vertically down the health readings the user can identify the toolset with the lowest health. By repeating these actions on the toolset the user can identify the tool(s) responsible for the low health reading and act accordingly. This representation combines the advantages of the hierarchical browser with a matrix arrangement. Information relationships across levels of abstraction can be made explicit by using a common visual variable and spatial arrangement. The result is a display that allows us to hide detailed data behind higher level metrics but can also reveal information at multiple levels of abstraction by folding out the matrix.



Node	Availability	Health	Health History
▾Etch			
▸Toolset_WetEtch1			
▾Toolset_WetEtch2			
▸Tool_W1A			
▸Tool_W1B			
▸Tool_W1C			
▸Toolset_DryEtch1			
▸Toolset_DryEtch2			
▸Toolset_DryEtch3			

Fig. 10. Relating data to higher-level metrics

6.4.3 Identify Relationships between Views

The views offer two alternative ways of looking at the system. While they are non-orthogonal, certain relationships exist between them that are important for decision making. For example, the volume at an operation contributes or correlates to the volume at a toolset, so knowing the health of that toolset is helpful if manufacturing decisions are about to be made. Similarly, if a tool is about to be shut down for repair, it is important to know the health of the other tools in the toolset and the volume of work that is about to arrive.

We make these relationships explicit through the use of contextual highlighting. On selecting a functional area, the related operations in the process flow are highlighted (fig. 11). This allows the user to see how product levels are related to the health of that area. As a user drills down through the functional hierarchy, the number of related operations decreases making it easier to see this relationship. The flattened presentation of the process hierarchy is helpful as functional activities are scattered throughout the process. From the other perspective if the user notices a peak or a trough in the process flow, rolling over an operation will highlight the related functional area.

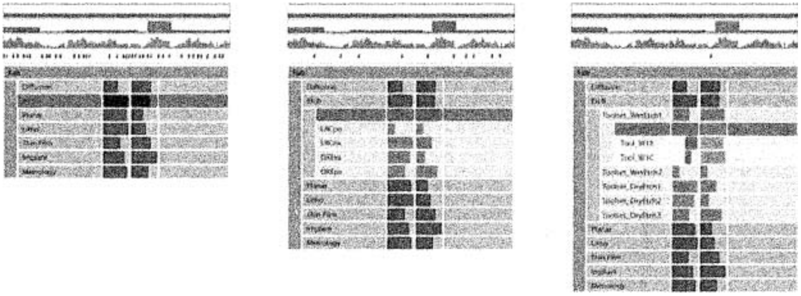


Fig. 11. Making relationships explicit

6.5 Make Relationships Explicit

To support Knowledge Based Behavior the user must be able to fully navigate both hierarchies. This allows them to gain a full understanding of the system state. This movement may occur in different ways depending on the task being carried out. It may be synchronous between levels of abstraction in both views, for instance comparing toolset health and product volumes at an operation. At other times information may need to be combined from different levels of abstraction as with the health diagnosis scenario discussed above (section 6.4.2).

A representation that displays multiple levels of abstraction allows us to visually jump between the metrics. This is the case with the process view where we can see product volumes at both operation and segment levels. Alternatively, interactive techniques can show contextual information on demand. The unfolding matrix in the functional view hides lower level data behind higher level metrics until they are requested. While these techniques allow us to navigate either hierarchy independently, the use of contextual linking could allow us to navigate them together making their relationship even more explicit.

6.5.1 Contextual Linking

We have already pointed out how contextual highlighting allows the user to identify relationships between views. Contextual linking would further extend this by allowing interaction with one hierarchy to control the display of the other. Above we describe how rolling over an operation in the process flow can highlight the related functional area. A contextual link would drill down into the functional view to the appropriate level of detail. For instance, when the user clicks on an operation in the process flow the engineering view would expand to display the related toolset.

This has further implications for the design of the process view. If the process flow operations are to be clickable there is a lower limit to the size that can be used to represent them. In the current design, area is used to encode product levels. However, at very low levels of production this area may be too small to act as a target. If we examine the activity being carried out on the display we can see that there is another option. Users are viewing the product volumes to note peaks or troughs in the data. Their task is not to estimate quantitative differences between adjacent operations but rather to understand the ordinal relationship between groups of operations. This is a simpler task and requires less detailed information. We carry out a scale transformation on the data turning detailed quantitative data into a series of ordinal ranges. Ordinal data can be supported by a wider select of visual variables including tonal value or shade [4]. By fixing the area of the operation and encoding the product volume using tonal value we can provide the user with a clickable representation of the process view (fig. 12) thus allowing for contextual linking.

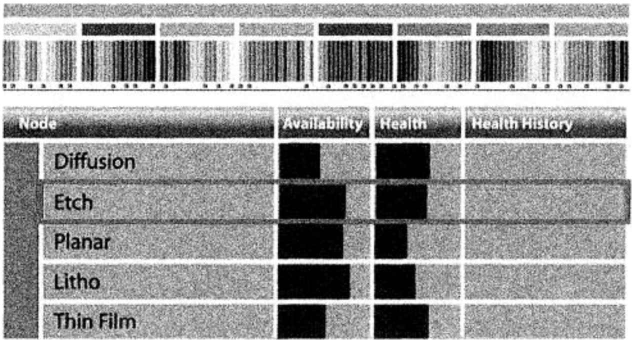


Fig. 12 View from final visualisation

7 Conclusions

EID provides us with a useful framework for designing interfaces for complex socio-technical systems. However, its application to large scale systems is not as straightforward as the approach given in the original framework.

In its first stage, the use of physical decomposition in the ADS is only useful where the physical constraints of the system match its process-flow. The introduction of circulation into process flows makes this approach unsuitable and requires functional constraints to be used instead. The manufacturing and engineering functional constraints associated with HVM are conflicting and cannot be integrated. Instead a modification of the ADS can allow us to integrate two abstraction hierarchies into an abstraction lattice.

In the second stage, we note that the "principles of visual display" discussed in section 2.2.2 are easily applied to smaller domains but less easily achieved with large complex domains. The scale of the Fab and the lack of a physical decomposition makes visual chunking unsuitable for embedding the ADS. Instead we must use more advanced visual techniques to represent our Abstraction Lattice.

Three major goals must be achieved to support KBB in the final interface. Firstly, both abstraction hierarchies must be explicitly expressed. Secondly, the hierarchies must be differentiable and compatible. Thirdly, cross-hierarchy relationships must be made explicit.

A design rationale has been produced which draws on the ADS to achieve these goals. Each hierarchy can be independently analysed under the terms of tasks, data types and structural depth. This information can be combined with guidelines for graphical data representation to inform the manner of hierarchical representation. Relationships between the two hierarchies can be established using a combination of visual organisation and interactive techniques.

8 Acknowledgements

This work has been funded through an Intel Ireland Research Scholarship. The Authors would like to thank Michael McGrath and Joe Butler at Intel® IT Innovation Centre, Intel Ireland, Leixlip, Co. Kildare, for their valuable assistance.

9 References

1 Woods, D., Johannesen, L, Cook, R., Sarter, N. (1994) Behind Human Error: Cognitive Systems, Computers and Hindsight CSERIAC

2 Zhang, J. & Norman, D. A. (1994). Representations in distributed cognitive tasks. Cognitive Science, 18, 87-122.

3 Hutchins E. (1995). How A Cockpit Remembers its Speeds. Cognitive Science, 19, 265-288

4 Bertin,J. (1983) Semiology of Graphics The University of Wisconsin Press, Madison. Translated by William J. Berg.

5 Zhang, J. (1996). A representational analysis of relational information displays. International Journal of Human-Computer Studies, 45, 59-74.

6 Zhang, J. (1997). The nature of external representations in problem solving. Cognitive Science, 21(2), 179-217.

7 Potter, S. S., Elm, W. C., Roth, E. M., Gualtieri, and J., Easter, J., (2002). Bridging the Gap between Cognitive Analysis and Effective Decision Aiding. In M. D. McNeese and M. A. Vidulich (Eds) State of the Art Report (SOAR): Cognitive Systems Engineering in Miltary Aviation Environments: Avoiding Cogminutia Fragmentosa! Wright-Patterson AFB, OH: Human Systems Information Analysis Center. (pp 137- 168). Also available at: http://iac.dtic.mil/hsiac/.

8 Vicente, K. J., and Rasmussen, J., "Ecological interface design: Theoretical foundations," IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-22, pp. 589-606, 1992.

9 Bisantz, A. M., and Vicente, K. J., "Making the abstraction hierarchy concrete," International Journal of Human-Computer Studies, vol. 40, pp. 83-117, 1994.

10 Vicente, K. J. (1999). Cognitive Work Analysis: toward safe, productive, and healthy computer-based work. Mahwah, NJ: Lawrence Erlbaum Associates

11 Rasmussen, J. (1979). On the structure of knowledge–A morphology of mental models in a man-machine system context. Risø Report M-2192. Roskilde, Denmark: Risø National Laboratory, Denmark

12 Rasmussen, J. (1985). The role of hierarchical knowledge representation in decision making and system management. IEEE Transactions on Systems, Man, and Cybernetics, 15(2), pp. 234-243.

13 Rasmussen, J (1983). "Skills, rules, knowledge: signals, signs, and symbols and other distinctions in human performance models." IEEE Trans. Syst., Man, Cybern. Vol. SMC-13.pp.257-267.1983.

14 Upton, C. and Doherty, G. (2005) Designing Usable Decision Support Systems for HVM. Proceedings of 10th IEEE International Conference on Emerging Technologies and Factory Automation, Vol. 1, pp. 459-466 Lo Bello, L. and Sauter, T.(Eds.)

15 Card, S., Mackinlay, J. and Shneiderman, B. (1999) Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann

16 Waloszek, G. (2004) "Fighting (with) Hierarchies – Part II: Presentation" SAP AG, Product Design Center – 04/27/2001, updated 01/29/2004. available at : http://www.sapdesignguild.org/community/design/hierarchies2.asp