RC23584 (W0504-046) April 11, 2005
Computer Science

IBM Research Report

Visualizing Supply Chains: A Design Study

Donna L. Gresh
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Visualizing Supply Chains: A Design Study

Donna L. Gresh [*]

IBM T.J. Watson Research Center

Abstract

Suppy chains describe the process by which materials are turned into products, and occur in a wide variety of contexts. As an example, consider the manufacture of desktop computers. Each computer is comprised of a variety of subcomponents, each of which is typically manufactured from a further set of components (and using various "capacities," for example, fabrication machinery). Each part has a cost and possibly limited availability; similarly each capacity has a cost and a maximum throughput. Sometimes one component may be allowed to be substituted by another, perhaps at a greater cost or with some lead time. The computers to be manufactured have projected demands as well as expected revenue when sold.

In the Mathematics department at IBM Research our particular focus is on *optimizing* supply chains, making recommendations on the best parts and capacities to use to produce a particular final product, and to predict shortages with enough lead time to do something to fix it. We can also recommend, based on limited available supply, which end products to manufacture, and when to manufacture them. As part of our asset base for solving such problems, we have the Watson Implosion Technology library, which allows one to model the supply chain, setting the relevant priorities, costs, constraints, etc. As one can imagine, real-life supply chains can be enormously complex, and there is a need for the optimization modelers to be able to visualize the model, as it is being constructed, to verify its accuracy and check for anomalies.

We have created a visualization application, which we call WitViz, which directly reads a supply chain model and presents an interactive, simple to understand graphical view of the supply chain, based on the Draw2D/GEF framework, after experimenting with alternative representation methods based on spanning trees. It allows the modeler to traverse the chain interactively, probe attributes, and see relationships quickly. The graphical representation closely matches the mental model that users hold about the relationships between objects in the supply chain. We also provide linked statistical views that allow users to see the range of node or link attributes, to look for outliers, or highlight particular values of interest. These highlights can also be fed back to the graphical representation.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

Keywords: visualization, information visualization, optimization

1 Introduction

The management and optimization of both internal and customer supply chains are an important business for IBM. For large, complex, manufacturing situations, the ability to model the supply chain, and respond dynamically to changes in it (for example, component shortages, demand changes, etc.) is critical to achieving profitability. The mathematics department within the IBM T.J. Watson Research Center has contributed to IBM's work in this area through the development of mathematical tools to model and optimize supply chains. For example, the management of the manufacturing of computer systems needs to take into account component supplies and costs, component substitution possibilities, demand forecasts, inventory costs, production costs, scrap value of unused parts, and a host of other variables. Both linear programming and heuristic models are used to find an optimal solution for manufacturing the computers, taking into account all of the attributes of the supply chain. (Heurstic models tend to be used in preference to linear programming models when it is difficult or impossible to specify quantitative values for some of the attributes. For example, practicioners are often able to specify a preference for the use of one part over another, but have great difficulty in assigning a monetary penalty to such a use. In such cases, heuristic models, which specify a preference, rule-based, ordering, can be helpful). These models are deployed in a wide variety of contexts beyond just the manufacturing of computers. For example, supply chain methods can be used to model staffing requirements, where one is interested in what the long-term needs for skilled personnel may be, given projected demand for such skills, and the possibility of geographic or other forms of flexibility in staffing. In addition, IBM operates a successful consulting business helping other companies model and optimize their supply chains. During the development of supply chain models, there is a need for modelers to be able to understand what they are building. Up until now there has been no way for the modelers to visualize the model they have built other than through "dumps" of the model as described by, essentially, "set attribute" methods. The application we have built allows the modelers to see the form of the model as it exists in-memory.

An outline of this paper is as follows: Section 2 discusses the requirements of the visual representation along with some of the characteristics of the typical supply chains we analyze. Section 3 discusses various approaches which have typically been used to analyze tree and network data. Section 4 dicusses the application as we have designed it, and shows some of the interaction mechanisms. Section 5 discusses the integration of more traditional information visualization views with the graphical representation, while Section 6 summarizes our contributions and suggests future directions.

2 Overview of the Problem

Our overall goal was to design a simple-to-understand visual representation of a "WIT Model," which would also allow interrogation of any attribute existing in the model. WIT stands for "Watson Implosion Technology," and is a software tool developed at the IBM T.J. Watson Research Center that aids in constrained materials management and production planning. Input to WIT is a list of demands for products, supplies of product components, and a multilevel bill-of-manufacturing, which describes how to manufacture a new component from a set of inputs. Typically, the list of demands for products is much smaller than the list of components required for these products. WIT "implodes" the list of supplies of compo-

[*]gresh@us.ibm.com

nents, via the BOM, into a relatively small list of feasible shipments of demanded products. Judicious trade-offs must be made between different demands, given limited supplies, to best satisfy manufacturing objectives. WIT is used internally at some IBM divisions, and has also been deployed at external customers around the world.

WIT models consist of parts, which may be of either material or capacity type (which simply indicates whether, if a part is not used in a given period, it is available for use in the next period, or is of a "use it or lose it" nature), operations, which create new parts from a set of materials and capacities, and demands for parts. In addition, WIT models contain BOM (bill of manufacturing) arcs which connect required inputs for an operation to the operation itself, and BOP arcs (bill of products) which specify the list of output products from an operation. Each component of a WIT model has a large range of possible attributes describing it, which allows a wide range of possible scenarios to be modeled. For example, parts can have a stock cost (inventory cost), a scrap cost (cost to get rid of them), a supply volume, an indication of whether a part can be shipped late, along with a dozen or so other attributes. Operations can have an execution cost, an incremental lot size (indicating whether fractional output is allowed), etc. BOM arcs may specify substitutes, indicating that it is allowed to use a different part or capacity, perhaps with a time or financial penalty, or a preference ordering. BOM arcs may also specify that a part is required in an earlier (or even later) period than an operation is executed, to model extended periods of manufacture time. Demands may be associated with financial rewards, which may depend on when the part is shipped.

Figure 1 shows a simple WIT model for products available at a diner. We will use this simple model, or a variant of it, in this paper to simplify the task of understanding the supply chain visualizer for the reader. In Figure 1, the final "parts" are various sorts of food items (circles near the top of the diagram), each with a demand (diamonds). To create these food items, "operations" (rectangles) are necessary, each of which requires both material parts (food items) and a capacity part (a pan, which is dedicated to the making of the food item for some unit of time). In addition, the BOM arc connecting American cheese to the operation for making a ham and cheese sandwhich also specifies a substitute arc of Swiss cheese. However the BOM connecting American cheese to the operation for making a cheese omelet does not, indicating that Swiss cheese is not an acceptable substitute for this operation.

Not explicitly shown is that most of these elements have attributes which depend on time. For example, the model may conver 10 periods of time (for the purposes of the diner example, these might be 2 minutes each), and the demand for bacon and eggs may be one unit in the third period, two units in the fifth period, one unit in the ninth period, and zero otherwise.

Watson Implosion Technology would take into account all the demands, all the constraints (for example, that it takes 2 periods to make bacon and eggs, and that no other operation can use the skillet during that time) and determine the optimal sequencing and use of materials and capacities. It can also take into account the profit margins and supply constraints, and, if necessary, decide to make one end product in preference of another. (One would hope that in a diner, as well is in a manufacturing supply chain, this would not occur too often to avoid alienating customers!)

Conceptually, this is the mental model of a WIT supply chain. As much as possible, the WIT visualizer maintains the essential characteristics of this mental model to aid in understanding, as we will describe below.

3 Previous Work

Since a WIT model is a network, we will give an overview of various approaches to the visualization of networks. (An excellent survey may be found in [11].) In many applications for network visualization, for example, telephony, the goal is to see unusual patterns of connectivity, *e.g.*, points with more than the usual number of connections ([6],[7]). In addition, there is often a real spatial (geographic) relationship between nodes and where they are drawn in the visual representation. In contrast, for our application the desire is not necessarily to see a global view of the network (which is already known to be highly dense in some areas; that is, that some parts are used in a multitude of different operations) but rather to be able to see the datails and the relationships in the model and confirm that they match what was planned. In fact it is critical that *every* detail be accessible in order to verify the model. Also, in our application, there is no natural spatial meaning to the positions of the nodes needing to be expressed, though there is a natural "parent/child" relationship between nodes, and final products are typically thought of as being at the top of the network, while parts which are not produced by any operation (raw materials) are typically thought of as being at the bottom, as has been drawn in Figure 1.

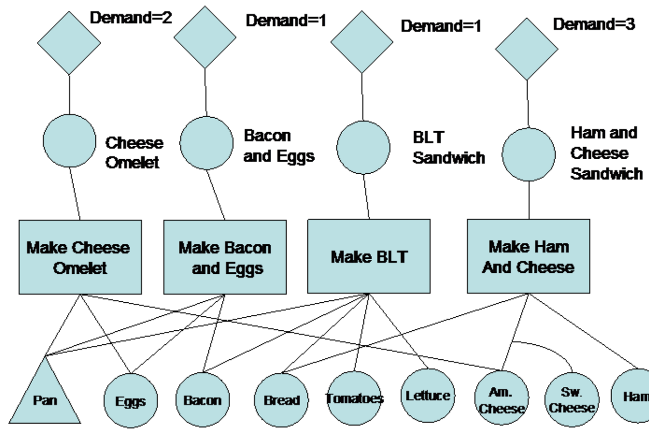


Figure 1: A simple WIT model to illustrate a conceptual visual representation using diamonds (demands), circles (material parts), triangles (capacity parts), operations (rectangles), BOP (bill-of-parts) arcs (upward lines from operations), BOM (bill-of-material) arcs (downward lines from operations), and substitute arcs (curved line). There are also a wide variety of possible attributes for each of these components, not shown in this representation.

Some graphs have attributes which simplify (or complicate) their graphical presentation. For example, for graphs which naturally have some degree of clustering, visualization can exploit that clustering in laying out the graph, as described by [8]. "Small world" graphs exhibit clustering, but also exhibit long-range connections between nodes in different clusters, significantly complicating their presentation, as described in [19].

A completely different approach to graph visualization is presenting the graph as a matrix, of size N^2 where N is the number of nodes, and the presence of a connection between two nodes indicated by a positive value in the matrix [2]. Alternatively attributes of nodes may be presented in a similar manner, as described in [1].

Significant work has been done in visualizing trees, as opposed to networks. Trees differ from networks in having a hierarchical pattern. For example, [16] and [22] discuss methods for visualizing and navigating in hierarchical graphs. Our supply chains are clearly not trees, and a minimal spanning tree is of limited usefulness in understanding a WIT model, in which *all* of the interconnections are of equal importance. H3Viewer [14] attempts to fill this gap between networks and trees using a spanning tree approach to lay out the graph, with non-tree edges shown on demand. This approach is of use mainly when there is a way to choose the "best" parent for a node. A novel tree comparison technique was described by [15]. We did experiment to some degree with tree-oriented visualizations, as we will describe below. In the Discussion section we will describe the relative pros and cons of our solution with other work.

There has been some previous work in the specific area of supply chain visualization. For example, the Supply Chain Visualization project at MIT ([17]) uses a tactile system to allow users to build their supply chain in a truly physical sense. This application is quite different from the complex supply chains our department typically deals with. Their project is more appropriate for allowing users to understand the mechanisms of supply chains in a high level way, but is not realistic for our problems. [23] describes a supply chain visualization system that also uses rather concrete metaphors to display components of a supply chain in a virtual reality framework. Again, this level of realism is impractical for the sorts of supply chains we deal with.

4 The WIT Visualizer

The simple example of Figure 1 uses the standard shapes and arrangements which WIT modelers use in drawing ideas on a white board. One design decision was whether or not to try and maintain this overall character in the WIT visualization.

An early prototype we created was to use a tree visualization paradigm. While the WIT model is not a tree, but is rather a network, our first version used the TreeView available as part of Microsoft's standard GUI components (but of course also available in very similar form in Java Swing or SWT). We started with this paradigm because the TreeView is familiar to users, allows simple iconographic labels for rapid understanding, and deals naturally with situations in which the number of subitems may be quite large. (In the case of a network drawing, along the lines of Figure 1, we were concerned that layout problems would be severe for large models). In addition, it allowed an easy way to label items in the tree with the attributes, through a context menu accessible through a right-mouse click.

Typically, the modeler does not want or need to see the entire network at a time, but rather needs to see the "inputs" and "outputs" for a given part or operation under consideration. Even using a fisheye [18] or hyperbolic [13] view of the network is unlikely to be of help here, simply because the network is likely to have far-flung, criss-crossing connected nodes; these techniques tend to be more applicable for trees than for networks, although [19] did discuss a distortion-based approach to networks, as we will discuss in Section 6. Another complication is that for some operations, the bill-of-material may include an extremely large number of parts and capacities, and the possibility of substitutes adds another whole level of interconnectivity.

Our initial approach was related to the idea of creating a spanning tree of the network ([11]), however it differs in certain fundamental ways. First, we did not choose a single "root" node based on some objective criteria about the "goodness" of the resulting tree as in [3]. Rather we had a number of root nodes defined in a natural way by the WIT model itself, consisting of either all the "end products," or all of the "raw materials." Second, we handled the issue of multiple connections from a leaf to different parts of the tree by simply replicating the leaf. All of each leaf's attributes were accessible from any place it existed. If the user wished to see where the leaf was connected, he or she can simply ask the visualizer for an "upward" view, which would be presented in a second window, paired with the first. We avoided an exponential explosion in the size of the tree by having an internal representation of only a small part of the tree at any one time.

In a WIT model, as described on a white board, for example, parts are shown as circles or triangles, for material or capacity parts, respectively. Operations are shown as squares or rectangles, and demands are shown as diamonds. This is as is shown in Figure 1. We felt strongly that a first requirement for a WIT Visualizer is to maintain these shapes, to reduce the cognitive overhead in understanding what was being shown. We did this with small icons attached to the nodes of the tree.

In order to represent the network nature of the WIT model using a tree, we presented two views of the tree, one looking "down," and one looking "up." This naturally fit with the typical tasks a modeller is engaged in. For example, one goal may be to determine why a particular end product is not being built. In order to understand this, one would drill down from that part to the operations which produce it. Context menus provide information on which operations produce this part, and which parts they require. The result was that it was relatively easy to navigate to find out, for example, that a particular part required to make the desired end product is in short supply. Similarly, another task might be to determine the effect of a shortage of a particular part or capacity. In this case, an "upward" view is useful in order to determine the cascade of outputs which depend on this part.

The initial prototype is shown in Figure2, where we are interrogating the attributes of the "Peppers," which are one of the components of an operation to "Produce Veggie Omelet," along with a skillet and mushrooms, plus the components of a generic "Plain Omelet" (presumably eggs and butter). BOM arcs, which have attributes of their own, are indicated by the black stick figures corresponding to each part. User reaction was lukewarm, due to the lack of corresondance to the mental model shown in Figure 1. We thus decided to revisit the idea of using a network drawing approach, with modifications as necessary to deal with more complex graphs.

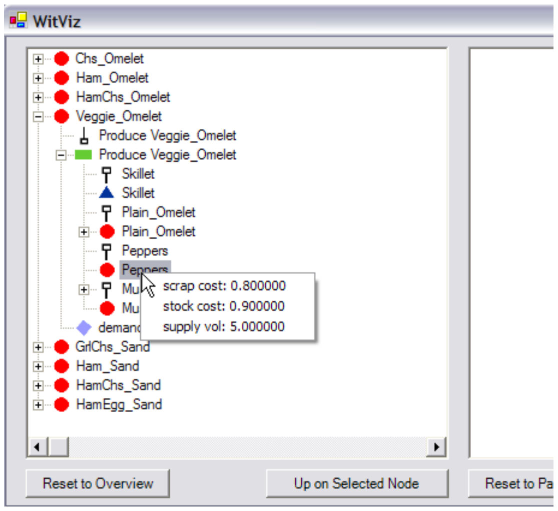


Figure 2: Detail of the initial WIT visualizer prototype. Information about a particular entry, in this case "Peppers," can be accessed through a right mouse click. Throughout this visualizer, there was a separation of object *attributes*, visible via context menus such as these, and object *connections* (*e.g.* parent/child relationships), visible via the tree hierarchy. The top level of the tree consists of the "final products" in the model, from a cheese omelet (Chs_Omelet) to a ham and egg sandwich (HamEgg_Sand). Not shown is the second, paired, window to the right of this one, which allowed the user to navigate "upward" (showing the outputs) of any selected item in this view.

We turned to an approach of using the Draw2D/GEF (Graphical Editing Framework) algortithms which are part of the Eclipse open development platform ([21], [20]).

For small models (which is often how the modeling process begins, with the understanding that the models are being developed programmatically, and thus are not necessarily always large) we recognized that we could begin by loading and drawing the entire model. The Draw2D layout algorithm creates a given sources and targets.

Some details that needed to be customized to match the WIT paradigm are that in WIT, sources are typically drawn at the bottom of the figure rather than the top, requiring that we redefine the interpretation of "source" and "target," and leading to some complications in dealing with the entrance and exit points of arcs. In addition, the typical WIT paradigm for drawing substitute arcs required the addition of invisible nodes to which to attach the arcs. However, overall, the Draw2D capabilities were well suited to the WIT models, and in fact the typical Draw2D layouts were eerily similar to what one would draw by hand. A typical small model is shown in Figure 3.

We build our graph as an instance of the class DirectedGraph in Draw2D, and apply a correponding layout algorithm to it. This algorithm lays out the nodes such that, as much as possible, source nodes appear above target nodes and edge crossings are minimized. Because the algorithm requires the graph to be fully connected (a requirement not shared by legal WIT models), we added invisible nodes and edges to complete the graph. All non-product parts (those not produced by any operation) are joined to one invisible node, and all operations with no BOM entries are joined to another invisible node. These two invisible nodes were then joined. Finally all demands are joined to another invisible node. The invisible nodes also had the desirable effect of pulling all demands toward the top and pulling all raw materials and BOM-entry-less operations toward the bottom of the graph.

One big advantage of this layout algorithm, over, for example, a force directed algorithm (which is commonly used to lay out networks) [5] is that it preserves the directed nature of a wit model: raw materials flow upward toward final products. While a force-directed approach could certainly attach arrows to indicate flow, the overall pattern of increasing complexity as parts are combined to make other parts would be difficult to perceive.

Figure 3 illustrates some of the characteristics and capabilities of the WitViz Visualizer. Context information pops up on a "mouse hover," in this case over with the mouse over an "operation" part, whose name might in some cases be too long to appear on the label. Each type of object is drawn with a graphical indication of its type: squares for operations, triangles for capacity parts, circles for material parts, and diamonds for demands (not shown in this figure). Colors are chosen to be distinct; red for operations, green for both capacity and material parts (which are more similar in a WIT model than they are different), and blue for demands. This particular WIT model contains a feedback loop near the bottom right of the image. In addition, this model has multiple BOM (Bill of Material) arcs from each "working" capacity to each engagement operation (near the top of the figure). (This particular model is modeling the assignment human resources to service engagements; an indication of the variety of application areas for which WIT is used). Multiple BOM or BOP arcs can occur in a multi-period problem, where a particular operation requires inputs over a span of time, rather than just one period, to complete its production. In this case it models a multi-month engagement which requires skilled personnel over a three-month period. It is possible that the quantities or other attributes can vary over those periods.

A design decision needed to be made as to how to handle such multiple arcs between the same two nodes. Our first approach was to avoid drawing multiple arcs between nodes, simply drawing

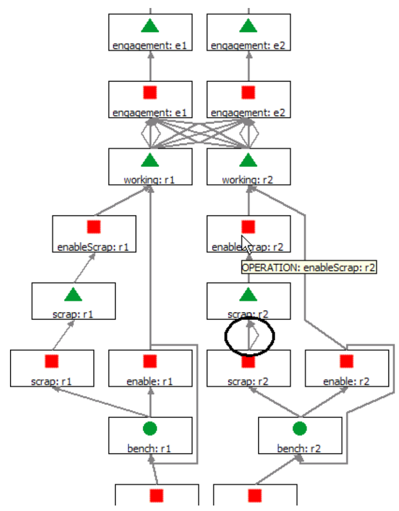


Figure 3: A typical small WIT model. Operations are indicated by squares, capacity parts by triangles, material parts by circles, and demands by diamonds (not shown). A mouse hover displays the full name of the object under the mouse. Note the multiple BOP arcs (circled); compare to Figure 4.

a thicker line to represent multiple arcs, to reduce screen clutter. However, we found that deviating from the conceptual model of the WIT model in this way was a poor design decision. It is perfectly possible for different BOM arcs to have different attributes. This was initially handled by simply listing the attribute information for all of the arcs represented by a single visual arc, in the "Details" text box shown in Figure 5. However, it is also possible for different BOM arcs between a single operation and part to have *different* substitute possibilities. We could think of no good way to show this graphically, other than by explicitly drawing the individual arcs.

We also found that visually, drawing multiple arcs was a more faithful representation of the data than a thick line representation, and in fact a model containing multiple arcs in an unexpected place was only noted to have multiple arcs once a switch to a multiple-arc drawing paradigm was made. Figure 4 shows a detail from earlier version of the visualizer which used a thick line to represent multiple BOP and BOM arcs. We did not even notice the multiple arcs connecting the "scrap: r2 operation" to the "scrap: r2 part" (circled) until we changed to a fanned representation of the arcs, as in Figure 3.

The full application window is shown in Figure 5. The left panel shows the full WIT model; the right half shows a "detail" view of the currently selected object (the pink highlighted "Veggie Omelet"), in a "Focus + Context" arrangement ([4]). The detail view shows the selected node, along with all of its children (inputs) and parents (outputs). This detail view is especially useful for larger models, where the inputs and outputs to a node may be widely separated. While the graph layout heuristic attempts to minimize the distances between connected nodes, this is clearly not always possible to achieve. Parent and child relationships are shown by darker blue colors for parents, and lighter blue colors for children. Highlighting the inputs and outputs in this way helps guide the eye to the relevant connections at the given moment, mitigating some of the inherent complexity in the graph. We also see in Figure 5 that the part "mushrooms" in the vegetable omelet allows several substitutes: eggplant, peanuts, and tofu. The fact that they are substitutes is indicated by a dotted line. We also see that details about the highlighted node are shown in the Details text field below the figure. The division between the "focus" and "context" windows, as

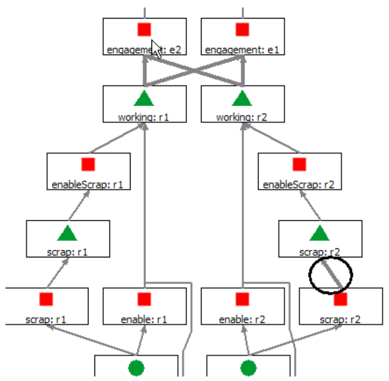


Figure 4: Detail of an earlier version of the WIT visualizer where multiple BOM and BOP arcs are indicated by a thicker line. The circled BOP arc(s) were not even noticed to be multiple until they were drawn separately; compare to Figure 3.

well as been the graph and the details windows, is movable by the user, thus allowing space to be used optimally. Figure 6 shows the view with the detail expanded fully.

As the user selects different objects in the "Full Model" view, the "Model Focus" view is updated to show the family of the current selected object (parents and children). At any time the user can also navigate in the detail view by selecting a node and pressing the "Refocus" button above (arrow symbol). This will cause a new detail family to be drawn, with the selected node as the new center of the family. In addition, the main view will be updated to bring the new focus node into visibility. (Simply clicking on a detail node, without requesting a "Refocus," will highlight the node in the main view and present attribute information about the node selected, but will not change the viewpoint, to avoid excessive context switching as a user simply queries information in the Model Focus view.)

We also recognize that, while the full view of the model is valuable, there will be times when it is impractical to show the entire model. In this case we offer the option to build a partial model of the data. In this case, the full model is read into memory, but the user is then offered a palette of, for example, parts or operations from which to choose a focus node. In this mode, the "Full Model" part of the canvas is not used, and the user navigates either by refocusing on selected objects, using the refocus, or arrow, button or by returning the part/operation list dialog to choose a new focus. We note that a searching for a new focus object is also available when using a full model view, which can be handy for finding a desired node quickly.

One can also iteratively expand on a node in the detail view to bring the selected node's parents and children into the display (using the expand, or "+" button). Figure 7 shows a model where an initial focus on the "Ham" material showed that part being a component in a number of operations, from Ham_Omelet to HamEgg_Sand. A click on the "Expand" button while HamEgg_Sand is selected results in the expansion of that operation to show all the parts it requires (from ham to egg) as well as well as what the operation produces (a ham and egg sandwich). Both the focus and expand options are available via double-click or right click mouse actions, respectively, for ease of navigation.

In a WIT model, edges have an importance equal to that of nodes, and carry a variety of attributes. For example, BOM, or Bill of Material, arcs may describe the number of items of a particular part needed by an operation, or the offset between when a part is necessary and the operation is completed. Of course the WIT visualizer must allow query of these attributes. In much the same way that a node may be selected, and its parents and children highlighted, an

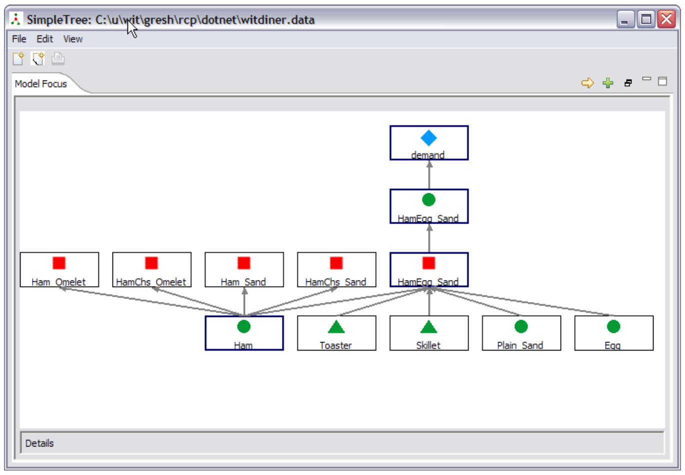
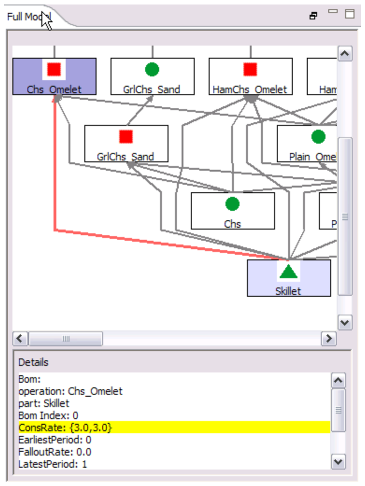


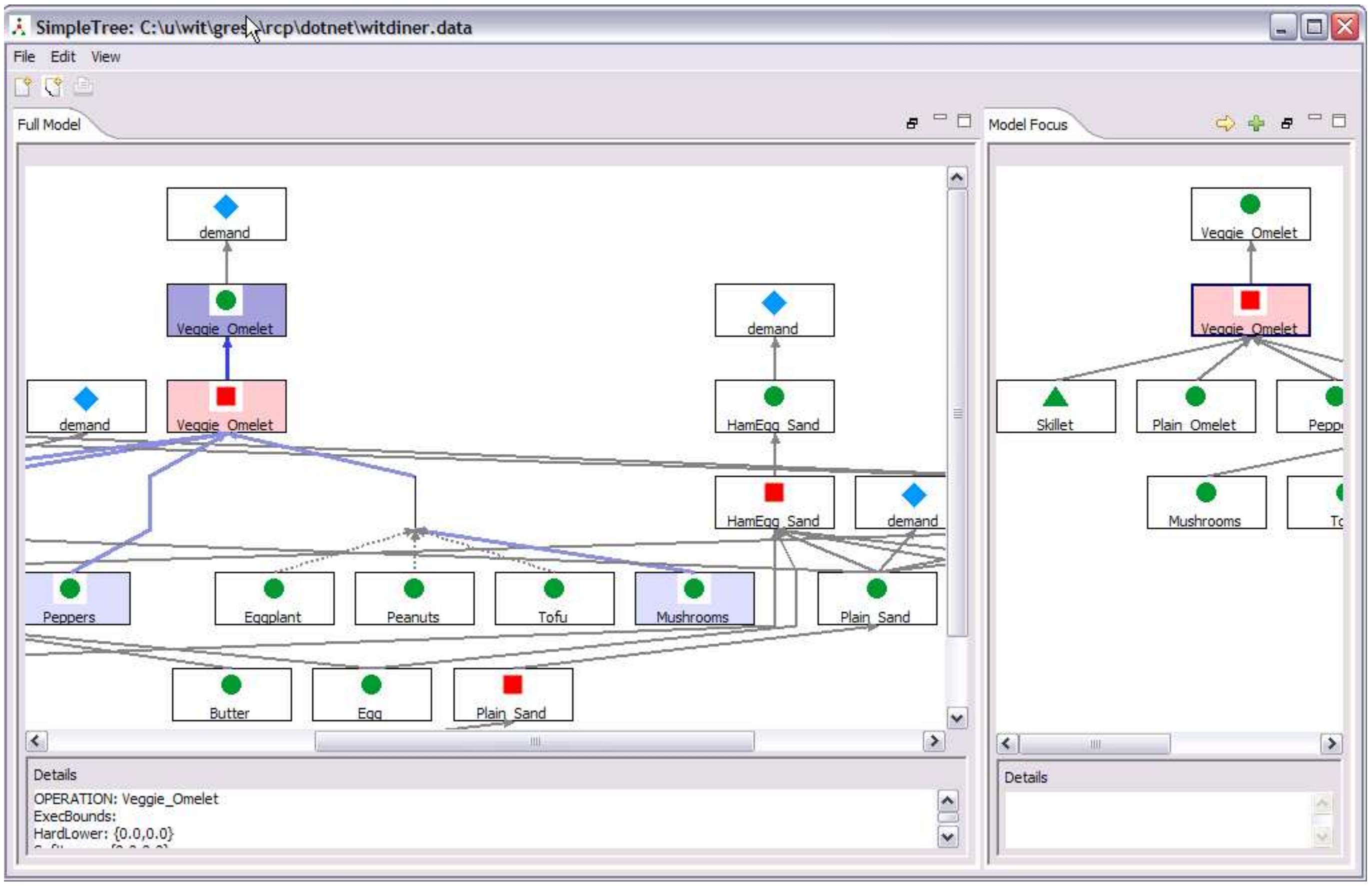
Figure 7: Iterative expansion of nodes allows exploration of graphs too large or unweildy to be drawn in full. Here, the neighborhood of Ham was requested first, then the user expanded the HamEgg_Sand operation, then the HamEgg_San part to see further details. This process can be repeated as many times as desired. Already expanded nodes are indicated with a darker border.

edge can also be selected, and its family shown. Figure 8 shows the result of selecting one of the BOM entries to the diner example. The source and target are highlighted in blue, analogously to selecting a node, and information about the BOM arc is displayed in the text box. In this example, the user has chosen to show all attributes, but to highlight those attributes which have non-default values. WIT allows a great deal of flexibility in modeling, with the result that typically, most attributes are defaulted, and generally the only "interesting" attributes are those which have been explicitly set. The user may also choose to show only non-default values.



```
Bom:
operation: Chs_Omelet
part: Skillet
Bom Index: 0
ConsRate: {3.0,3.0}
EarliestPeriod: 0
FalloutRate: 0.0
LatestPeriod: 1
```

Figure 8: Result of highlighting a BOM (Bill of Material) arc in the context window. The source and target of the arc are highlighted in a way analogous to selecting a node (see Figure 5). The detail window (not shown here) would also display the local family selected.



Details
OPERATION: Veggie_Omelet
ExecBounds:
HardLower: {0.0,0.0}

Figure 5: The WitViz visualization application window. The focus window (on right) or context window (on left) can be changed in relative size, or maximized or minimized to optimize screen real estate. Veggie_Omelet is the highlighted node, with its parents (outputs) shown in a darker color and its children (inputs) shown in a lighter color. A similar color scheme is used on the associated arcs. As is standard for WIT models, flow proceeds upward from initial materials to final products. Details of the attributes of the highlighted node are shown in the lower text box. The contents of the text box can be personalized by the user, to for example, show only non-default values. One of the inputs of Veggie_Omelet is Mushrooms, which allow substitute values of Eggplant, Peanuts, or Tofu. Substitute BOM entries are indicated by dotted lines.

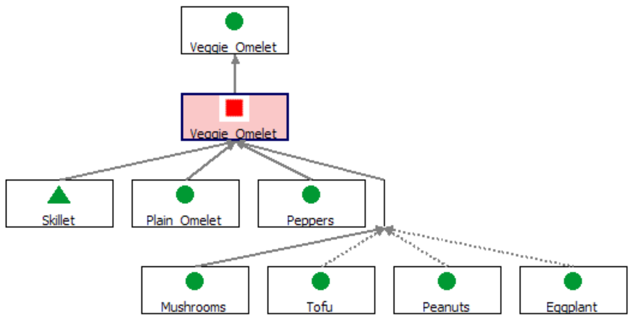


Figure 6: Detail view (focus) resulting from the selection of Veggie_Omelet in the context view (Figure 5).

5 Integration of Information Visualization Views

In addition to the graphical representation of the WIT model, we have also integrated information visualization views. These allow the user to explore the characteristics of attributes associated with elements in the WIT model. For example a histogram can be display the distribution of scrap costs for a part. This might be used to find outliers (for example, a part with a unreasonably large cost), or simply to focus on all parts of a price greater than a particular value. Brushing and linking allows the user to associate highlighted elements in one view with attributes in another view (such as an aggregate view akin to a pie chart). We used the Opal information visualization toolkit more completely described in [10]. Figure 9 shows the dialog for these views, where we have brushed in green all parts with a relatively large scrapping cost. We see the distribution of attributes of the highlighted parts in some of the other variables in this view, as well as in a complementary "proportional" view shown below (available on a separate tab of the dialog), which is particularly well-suited to categorical data. For example, we see that the highlighted, high-scrap cost parts are more represented in the second period of the model than in the first. In Figure 10, we have switched to another tab in the "Part Attributes" dialog which immediately shows us which parts have been colored; we have also chosen to highlight those same parts in the network view.

One design decision (though not an irrevocable one), was how much sophistication in brushing to expose. The Opal toolkit we are using allows brushing in multiple colors, with either mixed or partioned colors. We decided that for the current version we would allow brushing in a single color (green) rather than expose multiple brushing colors. We felt that this would simplify the learning process for this unfamiliar visualization technique.

6 Discussion

We have implemented a visualization application for WIT supply chain models using a fairly standard directed graph layout algorithm, combined with a Focus + Context window layout and brushing and linkage in information visualization views. The disadvantage of our graph drawing method is that for very large graphs (larger than about one hundred nodes), layouts become too complex to easily understand in an overall, global view, and may be very slow to lay out for graphs with more than several hundred nodes. In this case the user may switch to a mode of investigating the local neighborhood around any desired node, and iteratively expanding the nodes in the view. The big advantage of the presentation we chose is that it closely matches the characteristics of our users' mental model of a WIT model. Sources appear low in the drawing, and targets higher up; substitute arcs are drawn much as they are in a "white board visualization," and any attributes may be easily queried. An earlier representation based loosely on a spanning tree approach was disliked by users, who could not relate the presentation to their mental model.

Of relevance is whether a matrix representation would be appropriate to consider in this context. [9] performed a study comparing user performance using both a traditional "force directed graph" layout to the same graph represented using a matrix. They found that for graphs of size larger than approximately 20 nodes, the matrix representation allowed the tasks to be completed more quickly and with fewer errors. We note that the specific tasks assigned to the users were of a fairly generic nature ("estimate the number of nodes," or "find a particular link," or "find the most connected node.") While we apreciate the difficulty of designing good studies to measure user performance, we do note that some of these tasks are better accomplished separately from the graph presentation. For example, we explicitly compute the number of parts and operations, and present that information on demand, which seems

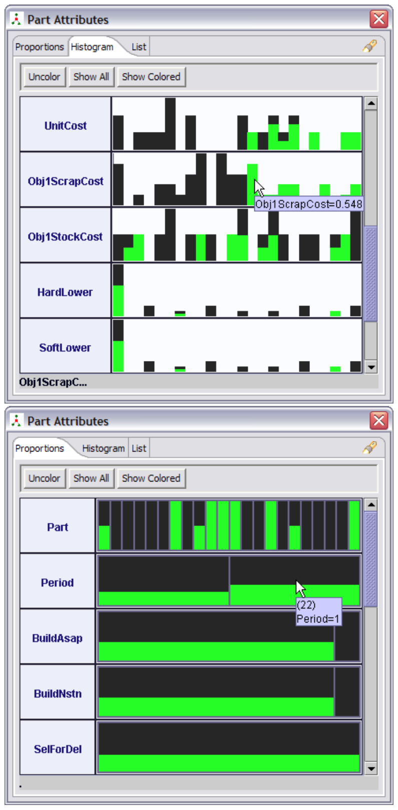
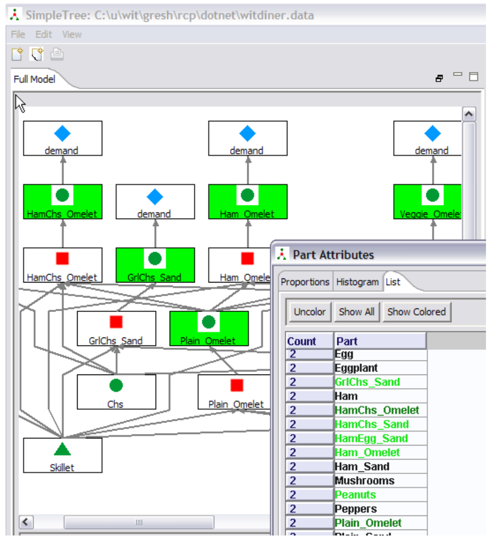


Figure 9: Information visualization views of the WIT model part attributes. Different linked views are presented as tabbed panels, two of which are visible here. Parts with relatively large values for scrap cost have been painted green. The "Proportions" view (lower) indicates that somewhat more of these high cost parts are found in the second period of the model.



Count	Part
2	Egg
2	Eggplant
2	GrlChs_Sand
2	Ham
2	HamChs_Omelet
2	HamChs_Sand
2	HamEgg_Sand
2	Ham_Omelet
2	Ham_Sand
2	Mushrooms
2	Peanuts
2	Peppers
2	Plain_Omelet

Figure 10: High cost parts are been brushed in the histogram view; here we see the names of the brushed parts, and we have also requested that the brushed parts be indicated in the graphical representation.

preferable to expecting a user to estimate this by eye. Similarly, the task of searching for a particular node (or link) is aided in our application by a search dialog that lists all elements (in a sorted list) and allows the user to zoom the graph directly to the selected element. (We note that in [12], where the authors investigated relative performance in understanding trees, as opposed to networks, using various visualization paradigms, that a "search" capability was one of the crucial elements missing in one of the techniques, causing much poorer performance on several tasks). The tasks which our users are engaged in typically relate to interrogating the attributes of elements, and seeing the relationship between elements; in this case we argue that having a more direct correspondence between the representation and the mental model is critically important. While a matrix representation would allow users to quickly find a particular node or link, and interrogate its attributes, moving up or down the supply chain from there would be highly non-intuitive, as compared to the same task using our approach. As one user put it, "Basically, I don't really understand a WIT problem unless I see a WIT diagram of it. I don't know any other way to comprehend the structure of a WIT problem. In the past, when users of WIT would ask me questions about WIT's behavior, I would ask them to provide a WIT data file of an example of a small problem (or problems) that illustrated the behavior in question. Sometimes they would also provide a WIT diagram. But since some of the users are in Japan and communicate with me by e-mail, they often only provide the data file and no diagram. In this case, historically, I would draw the WIT diagram myself, based on the data file. Not so easy to do: I can't predict the proper layout and of course, there's no way to be sure that I didn't get something wrong or leave something out. With WitViz, I get a reliably correct and well laid-out diagram very conveniently. This will be invaluable as time goes on."

We note that [19] explicitly addressed the problem of representing a graph that exhibited small world characteristics, using a variant of a force directed layout along with some distortion-based techniques to expose detail on demand. While an excellent improvement on standard force directed approaches for this class of data, it does not seem appropriate for our class of data, where a) the hierarchy, or "upward" flow is important, and b) clusters are not well-defined, and in any case are not "continuously zoomable" as they are in the small-world example. We note that our data sets are unlikely to be "small-world" in character (at least in the strict definition of the clustering index), if only because parts are never connected to other parts, only to operations, and vice versa. Thus, for example, an operation's children will never be directly connected to one another.

In the future, we would like to incorporate additional functionality to allow "debugging" of WIT models. For example, a simulation facility would allow stepping through a heuristic (rules-based) model time-step by time-step, and priority by priority in order to see how the rules influence the final result.

7 Acknowledgements

The author would like to thank colleagues J.P. Fasano, Bob Wittrock, Dan Connors, and Tom Ervolina in the Optimization and Mathematical Software group at IBM research for many helpful discussions and suggestions.

References

[1] James Abello and Frank van Ham. Matrix zoom: A visual interface to semi-external graphs. In *Proceedings Infovis, 2004*, 2004.

[2] J. Abelo and J. Korn. Mgv: a system for visualizing massive multidigraphs. *IEEE Transactions of Visualization and Computer Graphics*, 8(1):21–38, 2002.

[3] R.A. Botafogo, E. Rivlin, and B. Schneiderman. Structural analysis of hypertexts: identifying hierarchies and useful metrics. *ACM Trans. Information Systems*, 10(2), 1992.

[4] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers, San Francisco, 1999.

[5] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[6] Network visualization. Tutorial at IEEE Visualization, 2003. Stephen Eick, Chair.

[7] Stephen G. Eick. Aspects of network visualization. *IEEE Computer Graphics and Applications*, 1996.

[8] Yaniv Frishman and Ayellet Tal. Dynamic drawing of clustered graphs. In *Proceedings Infovis, 2004*, 2004.

[9] Mohammad Ghoniem, Jean-Daniel Fekete, and Philippe Castagliola. A comparison of the readibility of graphs using node-link and matrix-based representations. In *Infovis 2004*, October 2004.

[10] D.L. Gresh, D.A. Rabenhorst, A. Shabo, and S. Slavin. Prima: A case study of using information visualization techniques for patient record analysis. In *Proceedings of IEEE Visualization*, 2002.

[11] Ivan Herman, Guy Melancon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[12] Alfred Kobsa. User experiments with tree visualization systems. In *Proceedings, Infovis 2004*, October 2004.

[13] J. Lamping, R. Rao, and P. Pirolli. A focus and context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings CHI, 1995*, 1995.

[14] Tamara Munzner. Drawing large graphs with h3viewer and site manager (system demonstration). In *GD98: Symposium on Graph Drawing*. 1998.

[15] Tamara Munzner and Francois Guimbretiere. Treejuxtraposer: Scalable tree comparison using focus+context with guaranteed visibility. *ACM Transactions on Graphics*, 22(3):453–462, 2003.

[16] Quang Vinh Nguyen and Mao Lin Huang. A fast focus + context viewing technique for the navigation of classical hierarchical layout. In *Proceedings of Infovis 2003*, 2003.

[17] James Patten. Supply chain visualization. http://tangible.media.mit.edu/projects/scvis/scvis.htm, 2004.

[18] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In *Proceedings CHI, 1992*, 1992.

[19] Frank van Ham and Jarke J. van Wijk. Interactive visualization of small world graphs. In *Proceedings Infovis, 2004*, 2004.

[20] Eclipse, www.eclipse.org.

[21] Gef: Graphical editor framework, www.eclipse.org/gef.

[22] Jing Yang, Matthew O. Ward, and Elke A. Rundensteiner. Interring: An interactive tool for visually navigating and manipulating hierarchical structures. In *Proceedings of Infovis 2002*, 2002.

[23] H.B. Zhu, H.S. Tan, and H. Zhou. An internet database for supply chain visualization. In *The Proceedings of the International Conference on Integrated Logistics, 2001*, pages 123–128, 2001.