HAM: A GENERAL PURPOSE HYPERTEXT ABSTRACT MACHINE

BRAD CAMPBELL *and* **JOSEPH M. GOODMAN**

The HAM is a transaction-based server for a hypertext storage system. The server is designed to handle multiple users in a networked environment. The storage system consists of a collection of contexts, nodes, links, and attributes that make up a hypertext graph. The versatility of the HAM can be illustrated by showing how Guide buttons, Intermedia webs, and NoteCard FileBoxes can be implemented using its storage model.

The Hypertext Abstract Machine (HAM) is a general-purpose, transaction-based, multi-user server for a hypertext storage system. The HAM is based on the abstract machine Norm Delisle and Mayer Schwartz used in their Neptune system developed at Tektronix' Computer Research Laboratory [1]. Because the HAM is a low-level storage engine, it provides a general and flexible model that can be used in several different hypertext applications. For example, the HAM, combined with the Neptune user interface, provides a prototype system for a software engineering environment.

The HAM stores all of the information it manages in graphs, or databases, on a host machine's file systems. Graphs are stored in a centralized area and can be accessed in a distributed environment. The HAM typically communicates with an application through a byte stream protocol (although it may alternately be statically linked with an application). The application may or may not run on the same machine as the HAM.

Applications normally communicate with the outside world through a common user interface. The interface should be window-based and highly interactive to provide a suitable environment for a hypertext system.

Figure 1 shows the typical organization of a system using the HAM.

HAM FEATURES

The HAM storage model is based on five objects: graphs, contexts, nodes, links, and attributes. The HAM maintains history for these objects, allows selective access through a filtering mechanism, and can allow for access restrictions through a data security mechanism.

HAM Objects

A graph contains contexts, nodes, links, and attributes. These objects are organized hierarchically and carry the following descriptions:

A *graph* is the highest level HAM object. It normally contains all of the information regarding a general topic, such as the information for a software project. A graph contains one or more contexts.

Contexts partition the data within a graph. Contexts can be used to support configurations, private workspaces, and version history trees [6]. Each context has one parent context and zero or more child contexts. When a graph is created, a root context begins the tree. A context which contains zero or more nodes and links does not depend on information contained in its parent context.

A *node* contains arbitrary data that can be stored as text or as fixed-length binary blocks. A node can be classified as archived, nonarchived, or append-only. When an archived node is updated, a new version of

the node is created using the new contents. Previous versions of an archived node can be retrieved. When a nonarchived node is updated, the previous contents are replaced by the new contents. When an append-only node is updated, the new contents are appended to the previous contents. Append-only nodes are useful for logging the actions performed by an application. A node's contents can be searched for the occurrence of user-specified regular expressions. The search mechanism allows all versions of a node to be searched. Nodes are related by links.

A *link* defines a relationship between a source node and a destination node and can be followed in either direction. A *cross-context* link relates two nodes in different contexts and is useful for sharing data between two contexts. The generality provided by link attributes allows application writers to define their own notions of link types or link end-point attachment schemes.

Attributes can be attached to contexts, nodes, or links. Attribute values can be strings, integers, floating-point numbers, or user-defined types. Attribute/value pairs give semantics to HAM objects. They can represent application-specific properties of objects or contain information that further describes an object. Attributes are also used in the predicates that are part of the HAM filters.

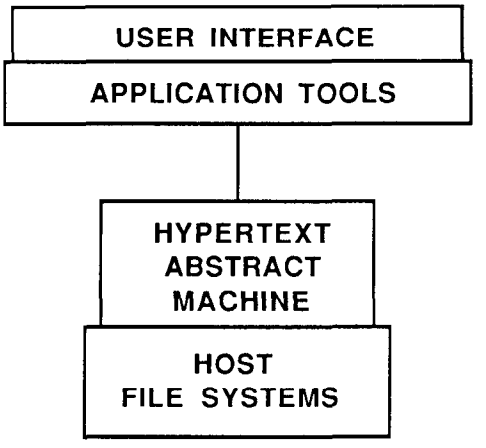

FIGURE 1. Generic Hypertext System Architecture

Version History

The HAM provides an automatic version history mechanism. The version history for a HAM object is updated each time that object is modified. Because each access to an object contains a version time, previous versions of objects can be viewed. The HAM also provides operations to destroy undesired versions.

Filters

The HAM provides a filtering mechanism that allows subsets of HAM objects to be extracted from large graphs. Filters allow the user to specify visibility predicates, which are expressions relating attributes and their values. HAM filters only return objects that satisfy the predicates. Filters also allow the user to specify a version time so that earlier versions of a graph can be examined.

The HAM filters the following items:

- Contexts in a graph
- Nodes in a context
- Links in a context
- Instances of a node in specified contexts
- Instances of a link in specified contexts
- A set of nodes and links in specified contexts based on a specific link ordering

Data Security

The HAM provides security for the data contained in a graph through its access control list (ACL) mechanism. Attaching an ACL to an object is optional. An ACL entry consists of a user or group name and a set of permissions. A user is anyone who has access to the graph. A group is a list of users. The available permissions are access, annotate, update, and destroy.

The permissions associated with an ACL entry are additive. Access permission allows the user or group to view the data associated with the object. Annotate permission allows links to be attached to a node. Update permission allows the user or group to perform nondestructive updates on an object. Destroy permission allows the destruction of an object.

HAM OPERATIONS

To provide a consistent, simple interface, HAM operations are grouped into seven categories. Operations within a category behave similarly, regardless of the object on which they operate.

Create operations create new HAM objects. A create operation takes object-dependent data and returns an object index and a version time. The object index represents a unique identifier for the newly created object, and the version time denotes the time at which the object was created.

Delete operations mark objects as deleted but retain historical information. A delete operation takes an object index and a version time, and returns a new version time. The object index specifies the unique identifier for the object being deleted. The returned version time represents the time the object was deleted.

Destroy operations free all space required for an object. The object does not have to be deleted to be destroyed. A destroy operation takes an object index and a version time, and returns a new version time. The object index specifies the unique identifier for the object being destroyed. The returned version time represents the time the object was destroyed.

Change operations modify data associated with an existing object. A change operation takes an object index, a version time, and object-dependent data and returns a version time. The object index specifies the unique identifier for the object being modified. The returned version time represents the time the object was modified.

Get operations retrieve data from existing objects. A get operation takes an object index and a version time, and returns the data that existed at the specified time. The object index specifies a unique identifier for the object from which data is being retrieved. The version time is a time range for the data retrieval.

Filter (and linearize) *operations* selectively retrieve information from a graph. A filter operation takes a predicate, a version time, and a list of attributes. These operations return a list of objects that satisfy the predicate and a list of requested attributes attached to each object. The version time specifies the time at which the filter is to search for the information. Each filter operation also has unique parameters in addition to those already specified.

Special operations are those that do not fit into any of these categories. They include functions such as searching for strings in node contents, merging contexts, and managing transactions.

EXAMPLE HAM APPLICATIONS

Because the HAM is a general-purpose hypertext engine, it can serve many types of hypertext systems. To illustrate this point we will model three hypertext structures using the HAM's storage model: Guide buttons, Intermedia webs, and NoteCards FileBoxes.

Guide Buttons

Guide is a hypertext product developed for the Macintosh by OWL International, Inc. of Bellevue, Washington [3]. It is a tool for writing and reading electronic documents. Guide uses *buttons*—special areas on a screen—to represent links in a document between the information on the screen and related information. When a button is selected, by clicking the mouse, Guide follows the link to display the related information.

Replacement buttons replace the button icon displayed on the screen with the information associated with that button. *Inquiries* are sets of two or more mutually exclusive replacement buttons. *Reference buttons* display the information associated with the button in a new window. This window remains visible until the user returns to the document window. *Note buttons* display information associated with the button in a new window that disappears when the user releases the mouse button.

To model Guide, the HAM equates a document with a node. The various button relationships are modeled as links. Link attributes determine which type of button the link represents. The application uses these link attributes to determine which type of window to open when a button is selected.

Figure 2 shows an example of a note button. The Document Browser contains the text being examined; the icon within the browser represents the note button. The Note Browser contains the note associated with the note button. The Button Attribute Browser shows the attributes associated with the link representing the note button, as well as the value of the *LinkType* attribute.

The button type is stored in the link attribute *LinkType*; its value is `Replacement`, `Inquiry`, `Reference`, or `Note`. All buttons also maintain the link attributes *Name* and *DocumentLocation*. *Name* represents the name associated with the button, and *DocumentLocation* defines the location relative to the beginning of the document where the button was created. The value of *DocumentLocation* corresponds to Guide's location of its button icon. Guide considers the information associated with a button to be an atomic entity. Therefore, the other end of the link representing the button can point to the entire node that contains the button's information.

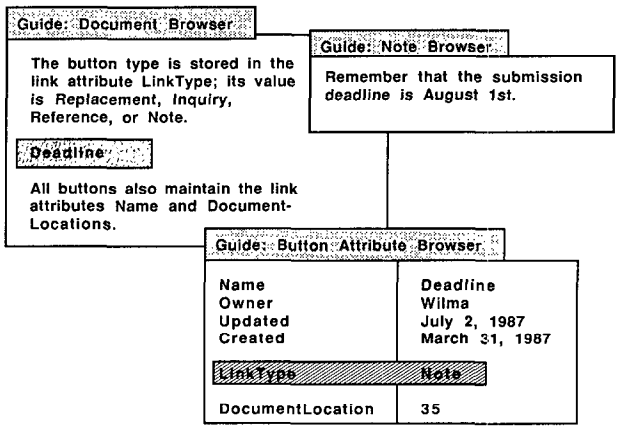

FIGURE 2. Possible Representation for a Guide Note Button

If a replacement button is part of an inquiry, the value of *LinkType* is set to `Inquiry`. A link that represents part of an inquiry also has an attribute named *Grouping*, which contains the identification of a special node. This node contains the identification of all links (replacement buttons) that make up the inquiry.

Figure 3 shows the HAM storage model for an inquiry named `Example Inquiry`. The Storage Representation window shows the nodes and links involved in the inquiry. In this example, the links have the same name as their destination nodes. The node `Example`

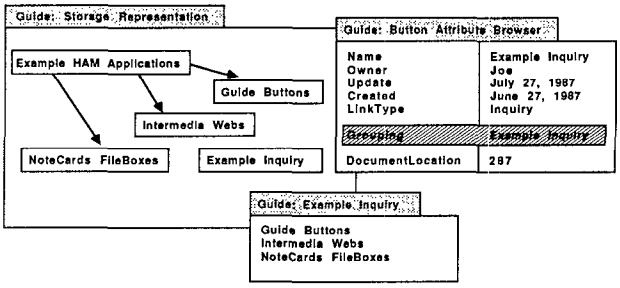

FIGURE 3. Inquiry Storage Representation

Guide is a trademark of OWL International, Inc.

Macintosh is a trademark of Apple Computer, Inc.

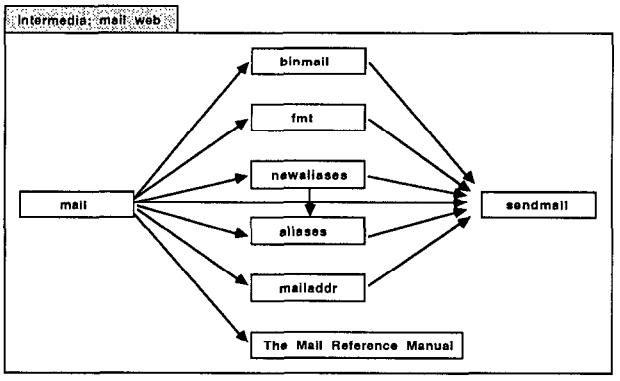

FIGURE 4. Mail Web

HAM Applications is the document node. The nodes Guide Buttons, Intermedia Webs, and NoteCards FileBoxes contain the information associated with the replacement buttons that make up the inquiry. The node Example Inquiry contains the names of the replacement buttons in the inquiry; its contents are shown in the Example Inquiry browser. The Button Attribute Browser displays the attributes attached to one of the links involved in the inquiry and shows that the value of the *Grouping* attribute is Example Inquiry.

Intermedia Webs

Intermedia, the system developed at the Institute for Research in Information and Scholarship at Brown University [2], [7], is one of the newer and more innovative hypertext systems.

The basic hypertext concepts in Intermedia are very similar to those found in the HAM. Intermedia uses the term *web* to refer to a database that contains both references to a set of documents and the links associated with those documents [5]. A *block* is the piece of a document to which a link is anchored and can be any legitimate selection in the application. The attributes provided by the HAM allow the flexibility to efficiently model these relationships.

To model an Intermedia web, the HAM represents a web as a collection of nodes and links. A document is represented as a node. An Intermedia link is equivalent to a HAM link. Blocks are determined by using link attributes to define the anchor selections for both the source and destination ends of each link.

UNIX manual pages[1] provide a convenient example of how the HAM can model Intermedia webs. The manual page for the mail command is used to create a small web of information.

UNIX is a registered trademark of AT&T Bell Laboratories.

[1] Excerpts from the UNIX Programmer's Manual, Berkeley Distribution, are used for purposes of illustration.

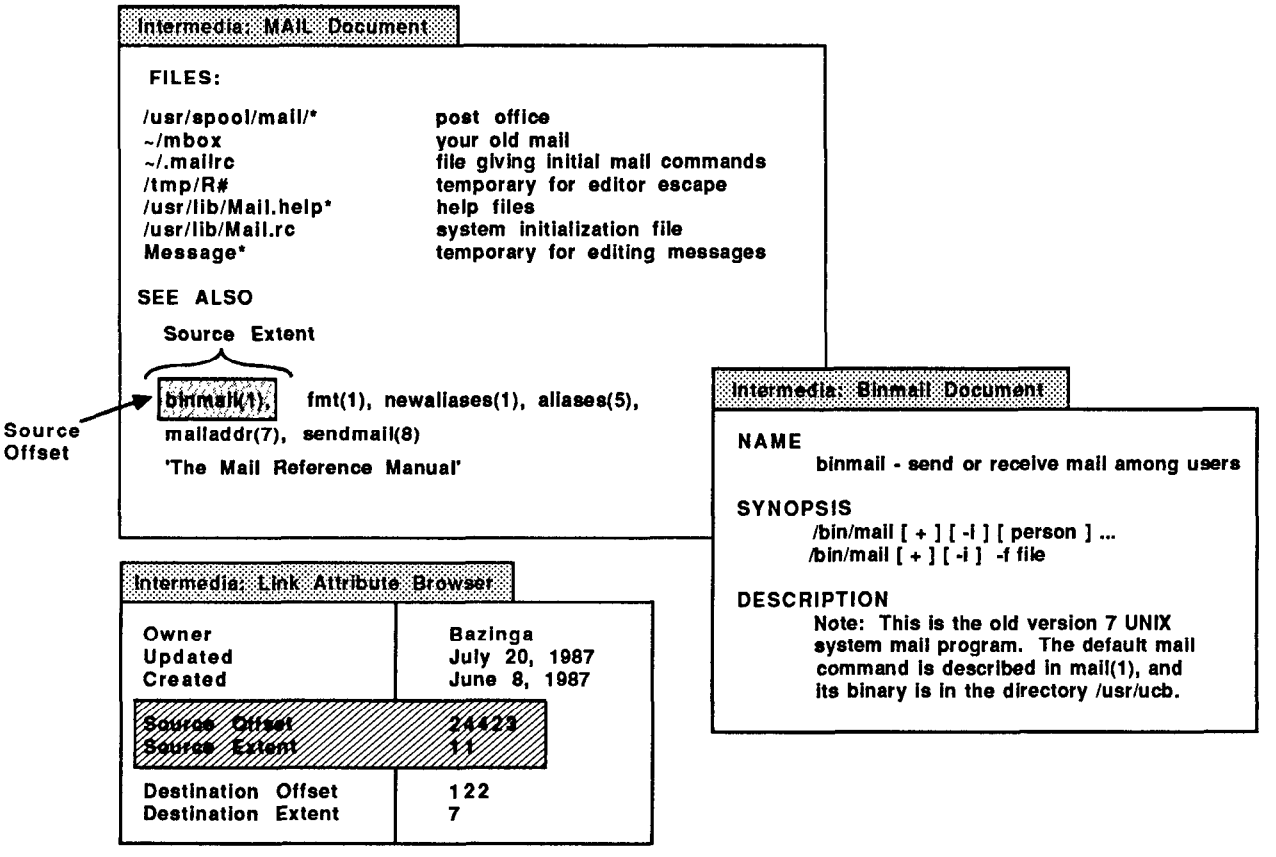

FIGURE 5. Defining a Block

Each document (manual page) is represented as a HAM node. The web is defined by attaching an attribute named *Web* to each link. The value of this attribute contains the name of the web to which the link belongs. A link filter is applied using the predicate "*Web* = `mail`" to let users view a map of the web. This filter returns only those nodes that are part of mail.

Figure 4 shows the mail web defined by creating links from the mail command to commands in the manual page's "SEE ALSO" section.

To define a block, the HAM uses the attribute pairs *SourceOffset/SourceExtent* and *DestinationOffset/DestinationExtent*. A block is determined by the value of the attribute pair attached to the link. For example, the source block of a link is represented by the attributes *SourceOffset* and *SourceExtent*. The values of these attributes are integers that contain the byte offset from the beginning of the node and the length of the block.

NoteCards FileBoxes

NoteCards is a general-purpose idea-processing hypertext system developed at Xerox PARC [4]. NoteCards supports the concept of FileBoxes. Every notecard must be stored in one or more FileBoxes. A FileBox which is arranged as a directed acyclic graph, can contain notecards and other FileBoxes.

FileBoxes can be represented in the HAM using nodes, links, and attributes. Both FileBoxes and notecards are equivalent to nodes. The model uses a node attribute to determine whether a node is a FileBox or a notecard. Links show which notecards (or FileBoxes) are in a particular FileBox. Link attributes determine which links refer to other FileBoxes and notecards. This model allows nodes to reside in more than one FileBox. The example shown in Figure 6 helps to clarify the NoteCards FileBox model.

The FileBox named `Hypertext 87` contains all of

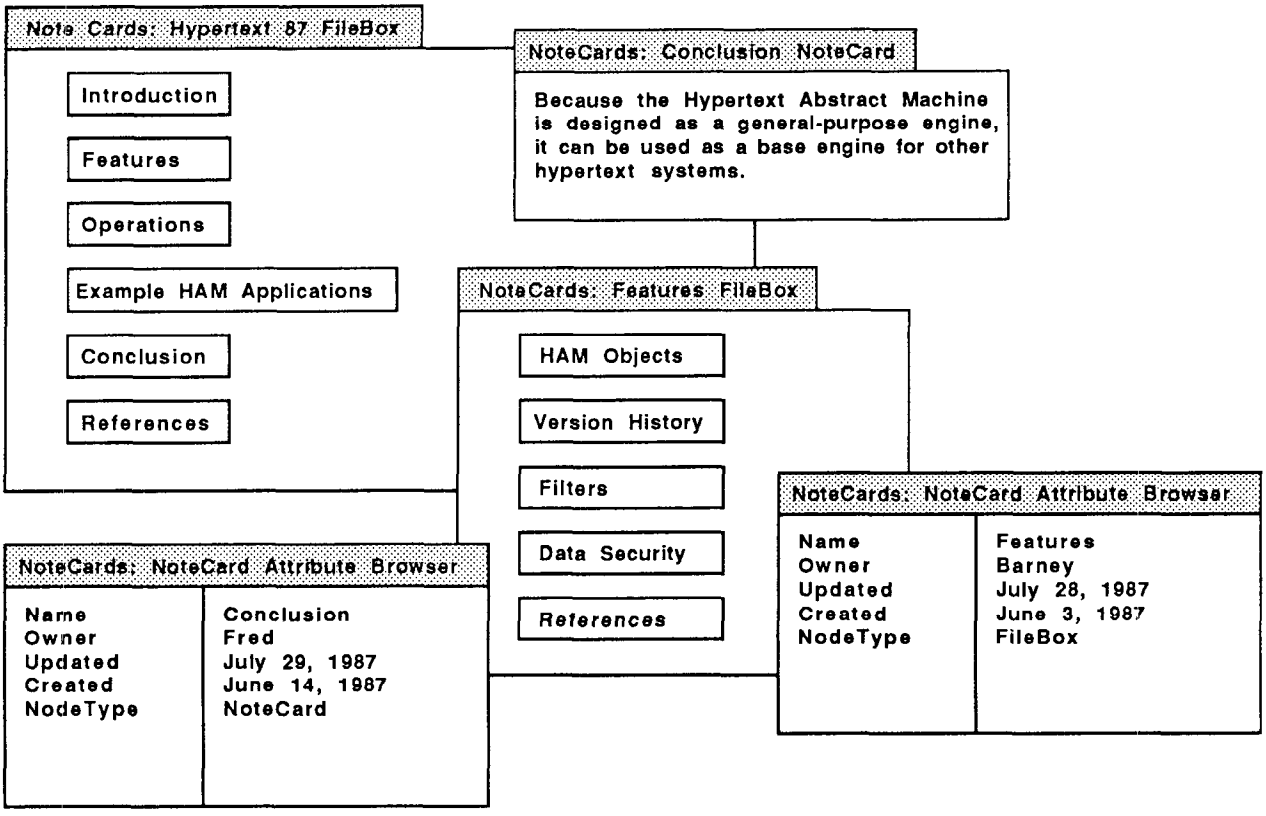

FIGURE 6. NoteCards Representation

Each block is defined by the offset and extent attributes. The offset provides an insertion point for the block, and the extent determines the end point of the block.

Figure 5 shows the value of the *SourceOffset* and *SourceExtent* attributes attached to link *BinMail*. The highlighted area shows the block these attributes define.

the FileBoxes and notecards that make up this article. As shown in the `Features` NoteCard Attribute Browser, the `Features` node is a FileBox. When a user browses this node, the NoteCards-like application examines the *nodeType* attribute, determines that the node is a FileBox, and opens a new FileBox browser. The contents of the `Features` node are links to all of the FileBoxes and notecards that it contains. It should

be noted that `References` is contained in both FileBoxes.

The `Conclusion` NoteCard Attribute Browser shows that the `Conclusion` node is a NoteCard. When a user browses this node, the application examines the *nodeType* attribute, determines that the node is a NoteCard, and opens a NoteCard browser.

CONCLUSION

Because the Hypertext Abstract Machine is designed as a general-purpose hypertext engine, it can be used as a base engine for other hypertext systems. Most current hypertext systems emphasize the application and user interface layers. While these layers are very important an appropriate storage model is essential. We believe the HAM provides such a model.

Although the HAM is not a panacea for hypertext data storage problems, it is an important first step. As new hypertext applications are developed, we will learn more about the data representation problems hypertext presents. If a storage model standard develops from this work, it may lead to the development of a standard terminology and base engine that could improve immeasurably the progress of hypertext technology.

Acknowledgments. We wish to thank Norm Delisle and Mayer Schwartz of the Tektronix Computer Research Laboratory for their helpful comments. We would also like to thank Victor Riley and Lee Thomas for their contributions to the HAM project. Lastly, we would like to thank Amy Rivero and Rich Davenport for their editing and illustration assistance.

REFERENCES

1. Delisle, N. and Schwartz, M. Neptune: A hypertext system for CAD applications. In *Proceedings ACM SIGMOD '86* (Washington, D.C., May 28–30, 1986), 132–142.
2. Garrett, N., Smith, K., and Meyrowitz, N. Intermedia: Issues, strategies, and tactics in the design of a hypermedia document system. In *Proceedings of the Conference on Computer Supported Cooperative Work* (Austin, Tex., December 3–5, 1986), 163–174.
3. *Guide: Hypertext for the Macintosh Manual.* OWL International, Inc., Bellevue, Wash., 1986.
4. Halasz, F., Moran, T., and Trigg, R. NoteCards in a nutshell. In *CHI + GI Conference Proceedings* (Toronto, Ontario, Canada, April 5–9, 1987), 45–52.
5. Meyrowitz, N. Intermedia: The architecture and construction of an object-oriented hypermedia system and applications framework. In *OOPSLA '86 Proceedings* (Portland, Or., Sept. 29–Oct. 2, 1986), 186–201.
6. Schwartz, M. and Delisle, N. Contexts—A partitioning concept for hypertext. *ACM Trans. on Office Information Systems 5*, 2 (April 1987), 168–186.
7. Yankelovich, N., Meyrowitz, N., and van Dam, A. Reading and writing the electronic book. *Computer 18*, 10 (Oct. 1985), 15–30.

CR Categories and Subject Descriptors: H.2.6 [**Database Management**]: Database Machines; H.3.4 [**Information Storage and Retrieval**]: Systems and Software; K.6.3 [**Management of Computing and Information Systems**]: Software Management
General Terms: Design, Management
Additional Key Words and Phrases: Abstract machines, hypertext systems

Authors' Present Addresses: Brad Campbell, CASE Division, Mentor Graphics Corp., 8500 S.W. Creekside Place, Beaverton, OR 97005; Joseph M. Goodman, Quantitative Technology Corp., 8700 S.W. Creekside Place, Suite D, Beaverton, OR 97005.