# Extraction of Structure and Content [123]

# from the Edgar Database: A Template-Based Approach

**Yu Cong**[4]

**Miklos Vasarhelyi**

**Alexander Kogan**

[1] This paper was accepted by Associate Editor Rajendra Srivastava.
[2] The authors are appreciative of the many useful comments of visiting editor Rajendra Srivasatava and two anonymous reviewers.
[3] The Edgar data can be obtained at http://edgar.sec.gov.
[4] Respectively, Assistant Professor Towson University, Professor Rutgers University and Professor Rutgers University.

# Extraction of Structure and Content from the Edgar Database: A Template-Based Approach

**Abstract:** This paper presents a template-based approach to extract data from the EDGAR database. A set of heuristic-based templates is used to configure the trainable system in order to have one type of EDGAR filings processed in a single configuration. Such configurability is highly desirable as it adds expendability and flexibility to this system. The template-based approach also enables the system to extract both structural information and content from the filings in the EDGAR database. The ability to extract structural information from a section or a complete filing makes it possible to collect data from real-world documents for users of financial data in both academia and industry. We use the income statement section of 10-K filings to illustrate the system and the utilization of the template-based approach.

**Keywords:** EDGAR, document structure, knowledge engineering

# INTRODUCTION

## Motivation

Advances of information technology such as XML are transforming the organization, storage and retrieval of information. This transformation inevitably changes the preparation, dissemination and use of accounting information. Document structure determines the understandability, accessibility and retrieval precision of a digital document (Fisher 2004). In the accounting domain, table-like text bodies (mostly financial statements) located in financial reports are the core vehicles of accounting information, and their structures are critical to the effective delivery of accounting information (e.g. Maines and McDaniel, 2000). However, without a thorough examination of the diversified structures of financial statements used in real world, the required understanding of relevant issues is impossible to gain.

Additionally, the development of digital accounting standards and languages such as XBRL also invites investigation into the structure of financial statements. For instance, Bovee et al. (2002) show that the rigid structure adopted by the first version of XBRL Taxonomy: Financial Reporting for Commercial and Industrial Companies – US GAAP (XBRL 2000) cannot accommodate the diversified structures of financial statements, particularly the income statement. Therefore, a thorough examination of the structures of financial statements used in the real world can help the accounting profession to gain the insight in the diversity of the structures of financial statements.

Moreover, the dramatic shift of standards for the XBRL taxonomy from rigid to virtually no structure reflects the fact that appropriate design of financial statements and reports in a digital format is extremely challenging. This fact in turn demonstrates the importance of a profound understanding of the structure of financial reports and the usefulness of such understanding to practitioners.

Research projects in accounting such as FRAANK aim at the extraction of accounting numbers but not their organization (structure) that includes grouping, sub-totals, etc. The extractions of structures in accounting research were typically carried out manually on small collections of financial statements (e.g. Bovee et al. 2002). Analyses on the structures of large collections of financial reports can examine the structural issues more thoroughly and thus add complementary evidence to the existing literature. However, such large-scale analyses are infeasible if not aided by computer programs that can automatically or semi-automatically extract the structural information from financial statements. In this paper, we contend that such computer aided extraction of the structure of financial reports is attainable, and attempt to design a system that accomplishes the extraction tasks by employing a template-based approach.

## The Tasks and Challenges

The technical difficulties of applying computer-aided analysis of the structure of financial reports is primarily posed by the source and format of the reports -- The Electronic Data Gathering, Analysis, and Retrieval System (EDGAR) [5] (U.S. Securities and Exchange Commission, 2003a 2003b) is maintained by the Securities and Exchange Commission (SEC). EDGAR is essentially the only free comprehensive source of electronic financial reports. Virtually all the analyses that require large number of financial reports [6] use the electronic filings from the EDGAR database or value-added tools based on EDGAR. Even though EDGAR has become the dominant source of financial reports to the general public, most of these financial reports are virtually unstructured free-form texts[7] -- a format that is extremely challenging for computer programs to parse and understand.

The extraction of the structure of financial statements and similar table-like text blocks must start with the location of these blocks in the financial reports. Such locating requires an understanding and extraction of the structure of the EDGAR filings. Only when the target block is located in the EDGAR filing, and the completeness and integrity of this block are preserved, the extraction of structural information from the block becomes feasible. When the structure of the table-like text block is extracted, the structural details such as the relationships between its line items and the sub-lists or sub-tables nested in the block must be captured. In addition,

[5] Gerdes (2003) provides a thorough review of the EDGAR database.
[6] EDGAR extraction provides "as reported" numbers as opposed to normalized data as delivered in COMPUSTAT.
[7] Companies are increasingly filing with EDGAR major financial reports in HTML format, but most filings are still in the free-form text format.

the extraction of content such as financial numbers becomes much easier when the structure of a table-like text block is extracted and understood. Therefore, two critical tasks in the structural extraction must be accomplished:

1) at the document level, to understand the structure of an EDGAR filing, locate the target table-like text block and extract the complete block with its integrity preserved,
2) at the statement/table level, to extract the structure and content of the block.

Although the two tasks seem to be trivial for a human expert, the same tasks are extremely challenging to automate. The first task is complicated by the great variability in the organization of the filings, the multiple presences of the same word(s) in multiple places, and the same concept expressed in several ways (synonyms) (Bovee et al. 2005, Kogan et al. 2001). Additionally, it is very common that one line item in a table-like text block is broken into several lines (multi-line parsing problem) and this factor complicates the structure of the block. In accounting literature, several studies address these challenges regarding the extraction of line items and accounting numbers (e.g. Bovee et al. 2005, Kogan et al. 2001, and Ferguson 1997). All these studies rely heavily on the heuristics of accounting knowledge. Additionally, the first task is challenged by the need for preservation of the integrity of the table-like text block.

The second task is also complicated by the issues above. Moreover, the structure of a table carries various types of formatting attributes ranging from white spaces to special characters. Furthermore, a large number of table-like text blocks such as financial statements typically have multiple sections and nested table structures. For instance, a multi-step income statement contains a number of sections such as revenue, cost, expenses, interest and tax expenses etc. Some of the sections may also be multi-level. For example, in Figure 1, Line21 to Line25 are components of Line27 and thus this section essentially is a table/list nested in the income statement table. Therefore, the extraction of the structure of table-like text blocks imposes more rigorous requirements on the use of heuristics, and thus the accounting heuristics alone are insufficient to accomplish the desired tasks.

===================

Insert Figure 1

===================

**Objectives**

In this paper, we attempt to address the challenges to the preservation and extraction of structures by the complete segregation of the logics for the two tasks, employing document structure models, and using a richer set of heuristics from both accounting and document structure analysis domains. In addition to the extraction of structural information, our system also extracts the contents (line items and financial numbers). Moreover, we use a template-based[8] approach that

[8] The term "template" refers to the encapsulation of multiple attributes/metrics into one container -- profile that is discussed in the system design and implementation section.

enables the system to be configurable and flexible. By changing the configuration files and repositories, the system can be configured to extract the data from the target table-like text blocks beyond major financial statements in EDGAR filings. We also carry out a sample study that configures the system to extract the structure and content from the income statement in 10-K filings. Our performance evaluation of this sample study renders high precision, recall and F-measure (all above 90%) in both tasks.

In the remainder of this paper we first review the relevant literature in accounting and document structure analysis. We then discuss the design principles, framework and implementation of the system followed by a sample study and evaluation of its results. Finally, we discuss the contributions to the literature and the accounting profession, the limitations of the system, and future work desirable to improve the system.

## RELATED WORK

### Document Structure Model

Except for Fisher (2004), the accounting literature rarely discusses the modeling of document structure. Fisher (2004) manually extracted data to examine the structures of financial accounting standards, used an XML DTD to model the structures, and made recommendation on how the structures should be improved. The objective of our study is to design a system that extracts structural information from unstructured or semi-structured documents and thus complements Fisher (2004). Models used in Document Structure Analysis (DSA) studies better serve our objectives.

DSA focuses on partitioning an electronic document into a hierarchy of physical components, a hierarchy of logical components, or both (Liang 1999, Wang 2002, and Klink et al. 2000). Tree-based document structure models are widely used in DSA studies (e.g. Liang 1999, and Klink et al. 2000). Such models are relatively simple to use but very desirable in the task of document structure extraction as they help to preserve the integrity of a logic block such as a table or list (Wang 2002). A typical document tree model depicts the structure of a document by two trees: one for the physical structure and the other one for the logical structure (Klink et al. 2000). Figure 2 visualizes the document tree model.

===================

Insert Figure 2

===================

Specific to the model of table-like text block, DSA studies are more focused on formalizing the procedures of: 1) table/block identification or background detection and 2) model the physical and logical layout of the table/block. Kornfield and Wattecamps (1998) develop a system that infers the structure of financial statements by using the logic tree model. Their system is essentially a parse-tree builder that extracts the content of the balance sheet and income statement to form a hierarchical tree structure. However, this system cannot locate the financial statements in financial reports. Douglas et al. (1995) and Douglas and Hurst (1997) adopt an approach that uses the relational model to model the underlying representation of a table. These models of table-like structure are insufficient as they are two-dimensional and contain various types of spacing and special characters. Hence, the use of the heuristics of physical and logical attributes to enrich the models is inevitable to the successful extraction of table structure.

**The Use of Heuristics**

DSA studies on table structures tend to use physical attributes to design domain-independent strategies for the modeling and extraction of table structures. Douglas and Hurst (1997) model the layout characteristics by "cohesions" that use attributes such as alpha-numeric ratio and string-length ratio to measure the "goodness of an area of a table." Pyreddy and Croft (1997) employ the alignment of white space as the critical physical attributes of tables. Ng et al. (1999) develop a prototype system that uses four classes of characters to model the physical structure of tables: 1) space character, 2) alpha-numeric characters, 3) special characters that are not in class one and two, and 4) separator characters that are one of ".", "*", and "%." However, the performance of all these systems suffers if domain specific knowledge is not provided.

Studies on the extraction of EDGAR filings typically use logical attributes and accounting knowledge to extract financial numbers rather than structures. Ferguson (1997) describes the development of the EDGARSCAN system (PWC Technology Center 2003) that uses several accounting heuristics to extract the financial numbers from the 10-K and 10-Q filings. The Financial Reporting and

Auditing Agent with Net Knowledge (FRAANK) (Kogan et al. 2001) also extracts financial numbers from 10-K and 10-Q. FRAANK maintains a repository of accounting synonyms and depends heavily on these to cope with the variations in the structure of the financial statements. In a later FRAANK version (Bovee et al. 2005) the extraction of financial statements and the extraction of items from these statements are segregated. No algorithm or model on the location and extraction of the statements is reported. The domain dependent heuristics appear to outperform domain-independent heuristics when financial numbers rather than table structures are extracted.

**System Design Strategies**

DSA and Information Extraction (IE) (Appelt and Israel 1999) use the same system design strategies. Similar to IE, the design of DSA systems is based on either a Knowledge Engineering Approach (KEA) or an Automatically Trainable Approach (ATA) (Appelt and Israel 1999). In a KEA based system, the logic is developed with the aid of human experts and these experts must be familiar with both the IE system and its knowledge domain. In an ATA based system, no human expert is required to train the IE system during its execution. When an IE system is first built, human experts who have domain knowledge need to train the system on what should be extracted.

The selection of a design strategy is fundamental as it affects the performance factors such as accuracy, speed, user-friendliness, etc. Essentially, the more heuristics are used, the more complex the system is and thus the more likely KEA is to be used. Moreover, a strategy/algorithm that decides on how the acquired knowledge should be applied to new documents is critical to the performance of a system, no matter whether KEA or ATA is used. Kornfield and Wattecamps (1998) adopt a "crawling" strategy that converts parse trees to templates and then maps data directly from the statements in one company's filing to those of the others within an industry. FRAANK (Kogan et al. 2001) uses an exhaustive search strategy to match between the term to be understood and the synonyms stored in a relational database.

## SYSTEM DESIGN AND IMPLEMENTATION

The design of our extraction system clearly segregates the logic that locates and extracts the target block from the logic that extracts the structure from the target block. This aims to attain a better preservation of structural integrity. In the logic that locates and extracts the target block, our design employs the physical and logical document trees used in DSA studies (Klink et al. 2000). In both approaches we integrate several types of physical and logical heuristics into one framework and thus take advantage of both the structural information and accounting domain-dependent information such as synonyms. Given the use of multiple types of heuristics and the complexity of the nested table structures of financial statements, we employ the KEA to take advantage of human experts' knowledge bases. We also adopt a "crawling" strategy adapted from Kornfield and Wattecamps (1998). However, our "crawling" strategy allows continuous addition and calibration of the templates previously generated and tried. We implement the entire system in Perl (Wall et al. 1996) for its superior capability of text pattern matching and processing.

**Architecture**

===================

Insert Figure 3

===================

Figure 3 presents the architecture of our design. The system consists of two segregated core components: Locator and Extractor. A *Data Preprocessor* fetches EDGAR filings from a Data Repository, cleans the filings and feeds to Locator. Locator identifies, locates and extracts the target text blocks and stores them in the Repository of Extracted Blocks and determines the heuristics for locating the table into the auxiliary repositories. Extractor fetches the text blocks from the Repository of Extracted Blocks, extracts the structures with the contents and stores them into the Repository of Normalized Extraction. The extractor also stores the heuristics in auxiliary repositories. A set of Knowledge Engineering Toolkits (KET) are provided to aid the users in validating the results and calibrating the heuristics. These components are discussed in more detail in the remainder of this section.

**Locator**

===================

Insert Figure 4

==================

*Locator* identifies, locates and extracts the target text blocks from an EDGAR filing and generates the heuristics of the document model that will be partly reused by the structure extractor. The core function of Locator, to identify the target text block, uses a cocktail approach that combines both the physical and logical attributes of the EDGAR filings. As shown in Figure 4, a *profile* is used to organize and encapsulate these attributes for each filing. The profile consists of four types of heuristics that are based on the document trees. Heuristics about the physical structure of a document include the geometric measurements of blocks such as the sparsity of a block and the structural attributes of blocks such as the left and right boundaries. The logic document tree-related heuristics include the semantics such as synonyms of the table titles and the text label of line items and the contextual relations between logical units.

==================

Insert Table 1

==================

Table 1 lists the attributes that are used in the document structure profile. Among these four groups of attributes, the contextual and semantic attributes play more important roles in identifying the target text blocks. Locator always attempts to use these two types of attributes first. The filings are first decomposed into logical units called *ITEMs* following SEC's requirement on the structure of a particular type of filing. Locator then searches through each logic unit for sub-units that may be the target text block by checking which sub-unit satisfies most of the attributes of all four types. When the target text block is located, the text block is stored into the Repository of Extracted Blocks. A *configuration slot* that consists of the absolute position of the starting and ending row in the document, and the absolute position of the starting row of each ITEM is also stored into the same repository.

In Locator, the processing starts with a sample pool of filings and "crawls" to similar filings iteratively. At the beginning of each iteration, the profiles for each individual filing in the sample pool are first provided by the user who is aided by KET. These profiles are then used by Locator as rules to generate attribute values from the other years' filings of the same company and companies within the same four-digit SIC-coded industry. The dissimilarities between the generated attribute values and the profiles are measured to decide if the new filing is correctly processed. The dissimilarities of the structural attributes are measured by whether they are identical or not, while the dissimilarities of geometric attributes are measured by the normalized absolute differences between the generated attribute values and the profile values.

If the processing is successful, a configuration slot is composed and stored and later on the target text block is extracted and stored. Since it is possible that the processing fails due to the lack of comparability between the structures of two companies' filings even if the two are in the same four-digit SIC group, Locator is designed to write the file name of the filing that failed the processing into log files. These logged filings are then re-sampled and the profiles based on the knowledge from user inputs are then added to the sample pool. After two iterations, it is possible that one 4-digit SIC group has multiple profiles. In the next iteration, Locator generates a set of attributes against each profile in the same SIC-coded industry and selects the set that has the least dissimilarity value.

**Extractor**

===================

Insert Table 2

===================

The architecture and logic flow of Extractor share some common traits with those of Locator. However, there are several differences between the two. First, Extractor uses a different set of attributes. Table 2 lists the attributes from the profiles used in Extractor. Most of the attributes are related to the pattern of a string, a row or a column in a table, though Extractor reuses some of the attributes, especially semantic ones, from the profiles generated by Locator. In fact, Locator and Extractor share the same repository of synonyms. Second, after the profiles in the initial sample pool are built, Extractor first generates profiles for the filings of the same year of the other companies in the same 4-digit SIC industry. This "crawling" heavily relies on the semantic and contextual attributes collected when the initial sample pool was built. User verification and calibration are performed at this stage following the same “generate-fail-learn” strategy that is adopted in Locator. Third, after one profile is generated for each company, Extractor "crawls" across filings of different years of the same company. In fact, in the actual execution, it has been found that very few modifications of the profiles are needed. Last, Extractor normalizes the extracted structures by replacing individual formatting with standard formatting. The normalized formatting attributes include indentation, spacing and the use of special characters. Figure 5 shows the normalized extraction.

===================

Insert Figure 5

===================

The critical sub-task in Extractor is to generate profiles for the companies that are not in the sample pool. We adopted an approach that relies heavily on logical attributes, especially semantics (synonyms). We further discuss the handling of synonyms in the Data Repositories and Synonyms section. In some cases, Extractor also tries to use the text label of line items, the relative position of terms and special characters (including white spaces) to locate table sections. The indentations are also measured throughout each section and replaced by a standard one generated by summarizing the measure of the indents of all the rows in the section. Finally, critical values such as the line numbers of the subtotals, position of column boundaries and special characters used are stored in the profile.

### Data Repositories and Synonyms

A number of data repositories are devised to store the data generated by different components. The system uses file-system-based repositories that write each profile or configuration slot in one file, store files in directories and use index files to search for and access the stored files. Two types of repositories are used in the system: main and auxiliary, as shown in Figure 3. The extractions, synonyms and some important intermediate data are stored in the main data repositories, while those used in heuristics, training and calibration are stored in auxiliary data repositories.

Due to the domain-specific nature of the system and the fact that there is tremendous variation of wording and phrasing in financial statements, this system faces a very challenging task while extracting composite phrases and matching synonyms. For instance, the system must be able to recognize "material, labor, overhead and direct cost” as a synonym of the standard term "Cost of Goods Sold." The semantic attributes in a profile are essentially synonyms of accounting terms. On the surface, only those terms used in the target text block are relevant. However, the system must also understand the synonyms that decide the textual attribute values in a profile. For example, to decide the relative location of a text block, Locator must have the knowledge of what terms are typically used to mark the beginning and ending of the text block. For example, in the case of an income statement block, to be certain that a text block that contains several "typical" terms is the "table of summary of selected financial data" rather than the income statement, the system needs to search for the balance sheet and the statement of cash flow items in the same block.

A file-system-based repository is used to store synonyms. The repository is initially fed with the results of a pilot study. The variations of a standard accounting term are stored in a file indexed after the standard term. For instance, "Cost" in Figure 1 is indexed by "Cost of Goods Sold." At the beginning of each execution, all the synonym files are loaded into a hash of lists in memory. During execution, Locator and Extractor retrieve exhaustively each hash when trying to set the semantic or some of the contextual attribute values. When the parsed text string and the stored synonyms are compared, string similarity measures[9] and treatments such as stemming are used if no exact match is found. In such a case, if the similarity is greater than 0.90 or the stemmed term can be exactly matched, the term is added to the synonym repository. If the term has already been placed in the repository, the count of matches is incremented by one. Expert aided verifications are conducted to check the newly added variations between iterations.

**Other components**

We adopt KEA (Appelt and Israel 1999) in our design and thus add a set of *Knowledge Engineering Toolkits* (KET) for two reasons. First, the logical attributes

---

[9] Based on the Levenshtein Edit Distance (Levenshtein 1966).

require the profound understanding of the accounting knowledge and thus invite the user's interpretation of the text blocks and terms. Second, the structural attributes can be very irregular and thus require a human expert's judgments. For example, some filers in health care industries may not include their annual financial reports in the 10-K filings named by their own central index key but include the reports in their parent companies' 10-K filings. Locator would be overwhelmingly inflated if logics that handle such exceptions are all added. Third, the size of the knowledge base is within the capacity of a human expert. We include in the KET two collections of scripts that are essentially the concise versions of the two core components of the system. These two are used to generate the profiles for the filings in the sample pool. Scripts that provide functions such as pattern matching, search, replace, compare, sort, merge, detection of duplicates, string analysis, word and character statistics, directory traverse and etc. are also included in the KET. These tools are used to clean, compare, verify and calibrate the synonyms and profiles stored in the repositories.

An agent-like module runs daily to retrieve the index files on the EDGAR database as well as to download and store both new data feeds and index files. A *data preprocessor* prepares inputs for the core logics in order to keep the core logics and the data clearly segregated. The preprocessor searches though the index files for target filings and extracts them from the data feed files. The preprocessor also strips headers, special attachments such as binary image files and confusing SGML tags from the filings and then feeds the cleaned filings into Locator.

## SAMPLE CONFIGURATION AND EVALUATION

A sample study on the 10-K filing and particularly its income statement blocks is configured and evaluated. The structure of income statement is one of the most diversified in all types of financial statements. We also implemented a *Structure Classifier* that classifies the extracted structures of income statement according to the AICPA classification system of income statement formats[10] (AICPA 2000) and writes the results into a repository of Classified Extractions. The addition of these two components assists the evaluation of the performance of structural extraction. We ran the system over all the 74132 10-K filings filed by 18455 companies and mutual funds between January 1994 and December 2002 available from the EDGAR database. There were 59557 extractions stored in the Repository of Normalized Extractions.

The performance evaluation was carried out in two stages: first to analyze how successfully Locator performed; then to measure the performance of Extractor. At both stages, in order to measure performance we used the commonly accepted methods such as *precision, recall,* and *weighted harmonic mean of precision and recall* (F-measure) (Baeza-Yates and Ribeiro-Neto,1999). These measures were initially used in information retrieval studies and then widely accepted by information extraction researchers. Since there is no benchmark system or generally accepted test data set, the evaluation does not involve a formal and complete comparison between our system and other existing systems.

[10] AICPA classifies the formats of income statement into four categories: 1) multi-step with gross margin reported, 2) multi-step without gross margin reported, 3) single step with income tax separately reported, and 4) single step without income tax separately reported.

To facilitate our discussion, Bovee et al. (2005) is referenced as a very rough benchmark.

In the performance measurement of Locator, *precision* measures how accurately Locator locates the income statement block in a 10-K filing by dividing the number of correctly extracted income statement blocks by the number of extracted income statement blocks ( P = the number of correct extractions/the number of extractions ). *Recall* shows the relevancy of extraction using the proportion of correctly extracted income statement blocks out of the 10-Ks that contain income statement block in the sample ( R = the number of correct extractions/the total possible extractions ). Usually there is a trade-off between the two measures. *F-measure* is a composite of precision and recall that balances the trade-off. The most commonly used F-measure gives equal weight to precision and recall ( F = 2PR/(P+R) ). In the evaluation, we require the extraction to be complete. Specifically, an extracted income statement block is deemed as correct only when the block preserves all the contents and format of an income statement block from the beginning of the first title line to the end of the last line, usually with special symbols such "=." We also examined the configuration slot to see if the line numbers of each 10-K Item were correctly recorded.

To analyze the performance of Locator, we randomly selected 750 filings (approximately 1% of all the 74132 filings). In the initial sample, 134 filings do not have matching extractions for several reasons. Of these filings, 113 are filed by filers that are not required to include financial statements in their 10-K or filed the statements with other filers in the same company group. Most of them are financial, real estate, health care and other service institutions. Twelve of them have irregular file formats as the filers filed several corrections or restatements and the SEC compiled these with the original filings. Five are either HTML or PDF files and contain contents that cannot be processed[11]. Four of them contain other irregularities that cannot be parsed. For instance, two filings contain only blank place holders for the income statement table. After removing these from the sample, 603 income statements were extracted from the remaining 616 filings and 584 of these were correct. Overall, the precision of Locator was 96.85% while recall was 94.81%. The F-measure was 95.82%. As a comparison, Bovee et al. (2005) reports the reliability (precision) of the table extraction logic of FRAANK on income statement to be 94% based on a test sample of 50 filings. The performance measures of Locator are summarized in Table 3.

===================

Insert Table 3

===================

In the second stage of the evaluation, we examined the performance of Extractor at the level of both statement and line items. At the statement level, we examined the correctness of the normalized extractions. On each normalized extraction we checked the following four aspects. First, we checked whether the

[11] For example, binary files such as picture or scanned documents are inserted into ASCII text. Such inserted objects hinder the pattern matching.

normalized structure was complete by comparing it to the original text block in the 10-K filing. Second, we examined whether the nested table/list was complete. For instance, the cost section in Figure 6 must preserve the label "Cost" and all the items under this label in the same order as in the original text. Third, we examined whether the terms used in the extraction were correctly indexed by the correct synonym. For example, "Cost" in Figure 6 must be indexed by "Cost of Goods Sold." Last, we examined whether the hierarchies were correctly represented in the normalization. For example, "Cost" is in the first layer while "Hardware Segments" is in the second layer. Out of 584 correctly extracted blocks, 551 (94.3%) were correctly normalized. We also examined the results stored in Repository of Classified Extractions to examine the correctness of the results and found that all the 551 correctly normalized extractions were also correctly classified.

At the line item level, *precision*, *recall* and *F-measure* were calculated in the same way as with Locator. Since multiple items were measured, we also calculated the measures of pooled items and the average of each measure on all the items. In the manual evaluation, we paid special attention to the integrity of the items. Specifically, we examined whether our system could correctly pick out items that spanned over two or more lines (multi-line parsing problem). In a case of multiple-line parsing, the extraction is labeled correct only if the whole item is captured and reshaped into a single line.

We used 603 documents extracted in stage one and selected line items based on the AICPA classification system of the income statement format (AICPA 2000). The line items we selected were those that could be used as classifiers to classify the income statements into the AICPA classifications. Therefore, we selected Revenues, Cost of Goods Sold, Gross Profit, Operating Income, Income Tax and Net Income to measure the performance. The precision was 97.55% while the recall was 96% when all the items are pooled. As a comparison, Bovee et al. (2005) reports the reliability (precision) of the combined parsing logic of FRAANK on income statement to be 91.3% based on a test sample of 50 filing. The results are reported in Table 4.

===================

Insert Table 4

===================

## DISCUSSION AND FUTURE WORK

This study presents the design, implementation and configuration of a template-based extraction system. This system employs structure models based on the physical and logical document trees. By using these models, we are able to integrate different types of document layout attributes into one "profile" and thus develop algorithms that take advantage of the structural and semantic information carried by these attributes. Even though our system, like existing ones, relies heavily on heuristics, and fragments of these heuristics can be traced to different previous studies, our "profile" integrates these heuristics in an organized manner and makes use of all of these heuristics. This cocktail approach helps to improve the system's ability to handle the nested table/list structures that are very typical in financial reports and other filings stored in the EDGAR database. Also, by combining both logic and physical attributes, the system is more capable of handling the problem of multiple occurrences of the same item in one filing and thus improves the performance of locating the target text block. Additionally, the encapsulation of attributes into a template improves the configurability and flexibility of the system and thus prepares the system for the extraction from the rich accounting related content in the EDGAR database.

Unlike most of the existing systems, our system completely and clearly segregates the logic of locating the target text block and the logic of extracting the structure and contents of the block. Such segregation offers several benefits. First, it enforces the integrity of the target text block and thus preserves the integrity of structure of the target text block. Second, a completely segregated target text block is less difficult to parse and thus the performance of the extraction from the block is improved. Finally, the two main components of our system are highly modularized and share similar logic flow with each other. This improves the extensibility of the system to the processing of other EDGAR filings. The architecture also segregates the logic from data storage and therefore improves the flexibility of the verification and calibration of intermediate data. Such flexibility is also very desirable as it makes the validation and calibration more domain-knowledge adaptive and thus improves the extensibility of the system.

The above two design features enable the system to address an issue that has not been adequately studied: to preserve and extract the structural information from table-like text blocks. The ability to address this issue has two contributions. First, it can help to recover the structure of table-like text blocks from the free-form text based EDGAR filings. The recovery of the structure of table-like text body domain adds to the accounting literature of document structure analysis (Fisher 2004) by extending the analysis to financial statements and tables. Additionally, by simply changing the configuration, the system can be used as an information extraction aid to researchers and practitioners. For instance, by configuring this system for the DEF 14A form, it can extract the executive compensation data and thus aid in the corporate governance studies.

Another immediate benefit of the structure recovery is to aid researchers and practitioners in understanding how financial statements are formatted and how the diversified formats affect the effective disclosure of accounting information. These understandings are useful to the accounting profession in three ways. First, the policy makers and standard setters can gain a better understanding of the relevance of the financial statement format to disclosure. Additionally, the designers of digital accounting protocols and standards, such as XBRL, can reference the formats when there is a need to build structures into the protocols or standards. Last, such understanding can assist various academic studies in the accounting domain. For instance, experimental studies in which the presentation formats of financial statements are varied (e.g. Maines and McDaniel, 2000) can be complemented by studies that make use of structural data extracted by our system.

The evaluation of the results from the sample study configuration shows that our template-based system is a potent research prototype that can be used in connection with applications where intensive parsing of the EDGAR filings in free-form text format is required. Moreover, the results also confirm that the cocktail approach used to construct the templates is effective. A rough comparison between the evaluation results of our system and those of FRAANK shows a similar precision between the two systems for the extraction of the financial numbers. However, very limited conclusions can be inferred from these numbers since the nature of the two systems and the testing scope and testing methods are different.

Our system is based on the KEA and thus requires the input of accounting knowledge from the users. Such a requirement limits the use of the system to experts. The most fundamental task of improving this system is to design a more robust algorithm that is based on a set of well-defined rules or statistics to rank the multiple groups of heuristics and select the most appropriate one “on the fly” at run-time. Such an algorithm would significantly reduce the amount of input from human experts and can eventually eliminate such input and transform this system into an ATA based system.

Our system has two technical limitations. First, the system lacks a user friendly interface. We currently use a command-line based toolkit. This interface is appropriate to the needs of a small group of users who know both accounting and Perl programming language. Additionally, the efficiency of the file-system-based data storage will decrease when the system is configured to process more types of EDGAR filings. These two limitations can be addressed in future studies by adding a GUI and using a relational database.

## REFERENCES


American Institute of Certified Public Accountant. 2000. *Accounting Trends and Techniques.* AICPA, New York.

Appelt D. E. and Israel D. 1999. Introduction to Information Extraction Technology: A Tutorial Prepared for IJCAI-99. *16th International Joint Conference on Artificial Intelligence.* http://www.ai.sri.com/~appelt/ie-tutorial/IJCAI99.pdf. Stockholm.

Baeza-Yates, R. and B. Ribeiro-Neto.1999. *Modern Information Retrieval.* ACM Press. New York.

Bovee, M., M. Ettredge, R. Srivastava, and M. Vasarhelyi. 2002. Does the Year 2000 XBRL Taxonomy Accommodate Current Business Financial Reporting Practice? *Journal of Information Systems* 16 (2): 165-182.

___________, A. Kogan, K. Nelson, R. Srivastava and M. Vasarhelyi. 2005. Financial Reporting and Auditing Agent with Net Knowledge (FRAANK) and eXtensible Business Reporting Language (XBRL). *Journal of Information Systems* 19 (1): 19-41.

Douglas, S. and M. Hurst. 1997. Layout and Language: Preliminary investigations in recognizing the structure of tables. *Proceedings of ICDAR 1997*, August: 18-20.

__________, M. Hurst and D. Quinn. 1995. Using Natural Language Processing for Identifying and Interpreting Tables in Plain Text. *Fourth Annual Symposium on Document .Analysis and Information Retrieval* 535-545. University of Nevada, Las Vegas.

Ferguson, D. 1997. Parsing Financial Statements Efficiently and Accurately Using C and Prolog. *Conference on Practical Applications of Prolog*. London, UK.

Fisher, I. E. 2004. On the Structure of Financial Accounting Standards to Support Digital Representation, Storage, and Retrieval. *Journal of Emerging Technologies in Accounting*. 1: 23-40.

Gerdes, J. Jr. 2000. Edgar-Analyzer: automating the analysis of corporate data contained in the SEC's Edgar Database. *Decision Support System*, 35 (1): 7-9.

International Business Machine Corp. 1999. *10-K filing (1998).* http://edgar.sec.gov/Archives/edgar/data/51143/0001047469-99-011848.txt.

Klink, S., A. Dengel and T. Kieninger. 2000. Document Structure Analysis Based on Layout and Textual Features. *Working Paper.* German Research Center for Artificial Intelligence, Kaiserslautern, Germany.

Kogan, A., K. Nelson, R. Srivastava, M. Vasarhelyi, and M. Bovee. 2001. Design and Applications of an Intelligent Financial Reporting and Auditing Agent with Net Knowledge (FRAANK). *Working Paper*. Newark, NJ.

Kornfield, W. and J. Wattecamps. 1998. Automatically Locating, Extracting and Analyzing Tabular data. 21st *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* 347-348. Melbourne, Australia.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-Doklandy*, 6: 707-710.

Liang, J. 1999. Document Structure Analysis and Performance Evaluation. *Doctoral Dissertation*. University of Washington, Seattle.

Maines, L., and L. McDaniel. 2000. Effects of Comprehensive-Income Characteristics on Nonprofessional Investors' Judgments: The Role of Financial-Statement Presentation Format. *The Accounting Review*, 75(2): 1-24.

Nelson, K., A. Kogan, R. Srivastava, M. Vasarhelyi, and H. Lu. 2000. Virtual auditing agents: the EDGAR Agent challenge. *Decision Support Systems*, 28 (3): 241-253.

Ng, H., C. Lim, and J. Koo. 1999. Learning to Recognize Tables in Free Text. *Proceedings of the 37th conference on Association for Computational Linguistics*: 443-450.

PWC Tech Center. 2003. A Technical Overview of the EdgarScan System. http://edgarscan.pwcglobal.com/EdgarScan/edgarscan_arch.html.

Pyreddy, P. and W. Croft. 1997. TINTIN: A System for Retrieval in Text Tables. *Proceedings of the Second ACM International Conference on Digital Libraries*: 193- 200.

U.S. Securities and Exchange Commission. 2000. HTML Tag and Attribute Specifications for EDGAR Release 7.0. http://www.sec.gov/info/edgar/ednews/edhtml.htm.

________________________. 2003a. Important Information About EDGAR. http://www.sec.gov/edgar/aboutedgar.htm.

________________________. 2003b. Form Types Used for Electronic Filing on EDGAR. http://www.sec.gov/info/edgar/forms/edgform.htm.

Vasarhelyi, M. and F. Halper. 1991. The continuous audit of online systems. *Auditing: A Journal of Practice and Theory*, 10 (1): 110-125.

Wall, L., T. Christiansen, and R. L. Schwartz. 1996 *Programming Perl*, 2nd Edition. Sebastopol, CA. O'Reilly and Associates.

Wang, Y. L. 2002. Document Analysis: Table Structure Understanding and Zone Content Classification. Ph.D. Dissertation, University of Washington.

XBRL. 2000. XBRL Taxonomy: Financial Reporting for Commercial and Industrial Companies, US GAAP. 2000-07-31. http://www.xbrl.org/Taxonomy/us-gaap-ci-2000-07-31.pdf.

_____. 2002. XBRL Taxonomy: Financial Reporting for Commercial and Industrial Companies, US GAAP. 2002-07-31. http://www.xbrl.org/Taxonomy/us-gaap-ci-2002-07-31.pdf.

**Table 1: The Attributes Included in the Document Structure Profile**

| Category | Attribute |
|---|---|
| Contextual | Relative position of logical units<br>Relative position of the income statement table<br>Relative position of terms (e.g., CGS appears after Revenue) |
| Semantic | The lead-in text (title, header, caption)<br>Text label of line items (usually Accounting Domain Specific, e.g., CGS)<br>Special characters (e.g., line feed and "=".) |
| Geometric | block sparsity (number of non-white character / total number of characters)<br>block numeric-alpha ratio ( number of numeric characters / total number of characters )<br>Length of the section |
| Structural | Left and right boundary of a block |

**Table 2: The Attributes Included in the Table Structure Profile**

| Category | Attribute |
|---|---|
| Contextual | Relative position of terms |
| | Sub-totals |
| Semantic | Text label of line items |
| | Special characters |
| Structural | Left and right boundary of a column |
| | Indentation of each row |

**Table 3: Evaluation of Locator Performance**

**Panel A: Sample Selection**

| | |
|---|---|
| Number of filings selected (approximately 1% of all the 74132 filings). | 750 |
| Filings do not contain income statements. | 113 |
| Filings with irregular file formats | 12 |
| Flings with unprocessable html or pdf formats | 5 |
| Filings with other irregularities that are unparsable. | 4 |
| Filings parsable. | 616 |
| Income statement extracted | 603 |
| Income statement correctly extracted | 584 |

**Panel B: Measurements**

| Item | Exists | Extracted | Correct | Recall | Precision | F-measure |
|---|---|---|---|---|---|---|
| Income statement | 616 | 603 | 584 | 94.81% | 96.85% | 95.82% |

**Table 4: Evaluation of Extractor Performance**

| Item | Exists | Extracted | Correct | Recall | Precision | F-measure |
|---|---|---|---|---|---|---|
| Revenues | 574 | 562 | 545 | 94.95% | 96.98% | 95.95% |
| Cost of Goods Sold | 383 | 368 | 360 | 93.99% | 97.83% | 95.87% |
| Gross Profit | 267 | 265 | 258 | 96.63% | 97.36% | 96.99% |
| Operating Income | 410 | 404 | 396 | 96.59% | 98.02% | 97.30% |
| Income Tax | 422 | 413 | 402 | 95.26% | 97.34% | 96.29% |
| Net Income | 595 | 597 | 584 | 98.15% | 97.82% | 97.98% |
| *Pooled* | 2651 | 2609 | 2545 | 96.00% | 97.55% | 97.98% |
| *Average** | | | | 95.93% | 97.56% | 96.73% |

* Average is the arithmetic average of Recall, Precision and F-measure

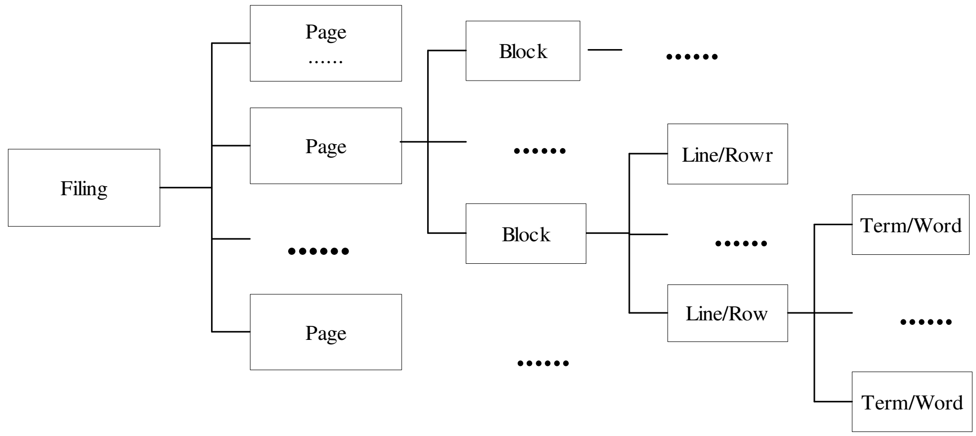
**Figure 1*: The Income Statement Section from IBM's 1998 10-K Filing**

```
         012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678
Line01:  CONSOLIDATED STATEMENT OF EARNINGS
Line02:  International Business Machines Corporation and Subsidiary Companies
Line03:  <TABLE>
Line04:  <CAPTION>
Line05:  - -------------------------------------------------------------------------------------------------
Line06:  (Dollars in millions except per share amounts)
Line07:  For the year ended December 31:                        Notes      1998        1997*       1996*
Line08:  - -------------------------------------------------------------------------------------------------
Line09:  <S>                                                       <C>     <C>         <C>         <C>
Line10:  Revenue:
Line11:  Hardware segments                                                 $35,419     $36,630     $36,634
Line12:  Global Services segment                                            28,916      25,166      22,310
Line13:  Software segment                                                   11,863      11,164      11,426
Line14:  Global Financing segment                                            2,877       2,806       3,054
Line15:  Enterprise Investments segment/Other                                2,592       2,742       2,523
Line16:  - -------------------------------------------------------------------------------------------------
Line17:  Total revenue                                                      81,667      78,508      75,947
Line18:  - -------------------------------------------------------------------------------------------------
Line19:  Cost:
Line20:
Line21:  Hardware segments                                                  24,214      23,473      22,888
Line22:  Global Services segment                                            21,125      18,464      16,270
Line23:  Software segment                                                    2,260       2,785       2,946
Line24:  Global Financing segment                                            1,494       1,448       1,481
Line25:  Enterprise Investments segment/Other                                1,702       1,729       1,823
Line26:  - -------------------------------------------------------------------------------------------------
Line27:  Total cost                                                         50,795      47,899      45,408
Line28:  - -------------------------------------------------------------------------------------------------
Line29:  Gross profit                                                       30,872      30,609      30,539
Line30:  - -------------------------------------------------------------------------------------------------
Line31:  Operating expenses:
Line32:
Line33:  Selling, general and administrative                       R        16,662      16,634      16,854
Line34:  Research, development and engineering                     S         5,046       4,877       5,089
Line35:  - -------------------------------------------------------------------------------------------------
Line36:  Total operating expenses                                           21,708      21,511      21,943
Line37:  - -------------------------------------------------------------------------------------------------
Line38:  Operating income                                                    9,164       9,098       8,596
Line39:  Other income, principally interest                                    589         657         707
Line40:  Interest expense                                          L           713         728         716
Line41:  - -------------------------------------------------------------------------------------------------
Line42:  Income before income taxes                                          9,040       9,027       8,587
Line43:  Provision for income taxes                                Q         2,712       2,934       3,158
Line44:  - -------------------------------------------------------------------------------------------------
Line45:  Net income                                                          6,328       6,093       5,429
Line46:  Preferred stock dividends                                              20          20          20
Line47:  - -------------------------------------------------------------------------------------------------
Line48:  Net income applicable to common shareholders                      $ 6,308     $ 6,073     $ 5,409
Line49:  =================================================================================================
Line50:  Earnings per share of common stock--basic                 T       $  6.75     $  6.18     $  5.12
Line51:  Earnings per share of common stock--assuming dilution     T       $  6.57     $  6.01     $  5.01
Line52:  ========================================================================================
```

*: The line and column numbers are added by the author to facilitate discussion.

**Figure 2: Document Tree**

**Panel A: Physical Document Tree**




**Panel B: Logical Document Tree**

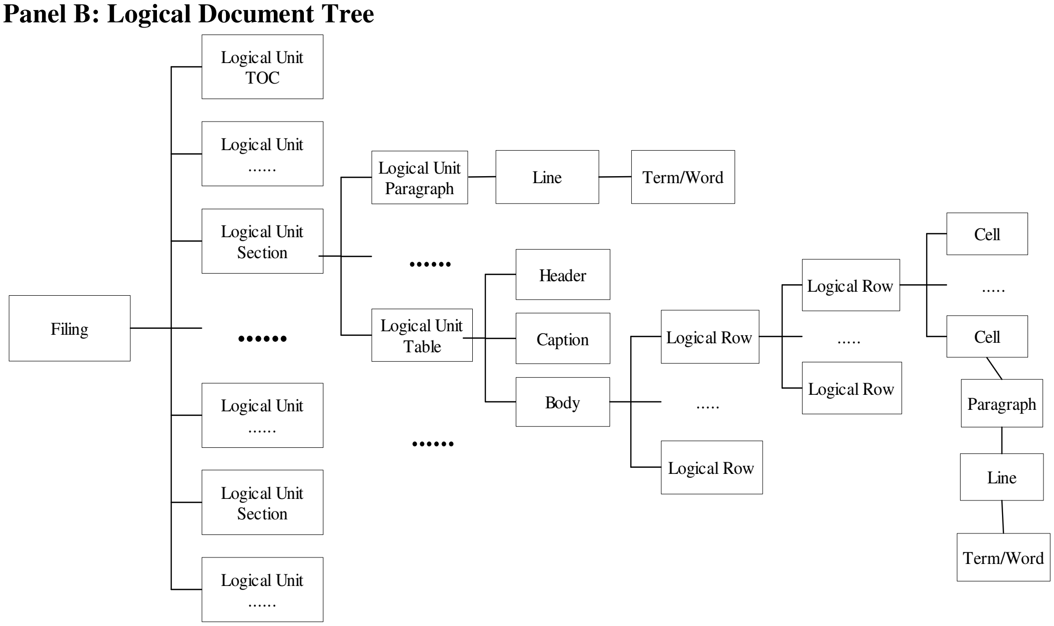
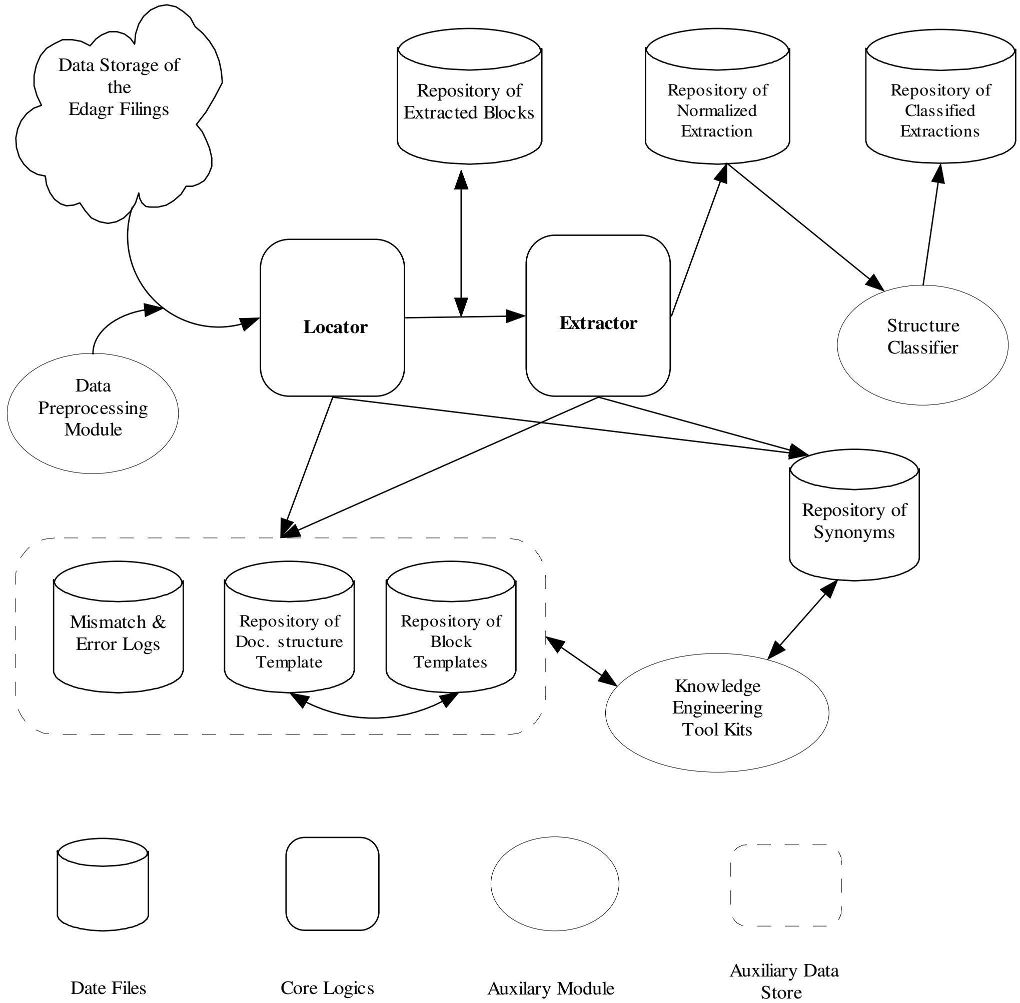


**Figure 3. The Architecture of the Extraction System ***




*. The Structure Classifier and Repository of Classified Extractions are used only in the sample study.

**Figure 4: Excerption of the Profile for HCA's 1995 10-K Filing**

```
Block Separator:
      \n\n;
Relative Position of IS:      ITEM 14
      After:            Regular Paragraph
      Before:           Balance Sheet
Lead-in Text:
      <               COLUMBIA/HCA HEALTHCARE CORPORATION>
      \s+27CONSOLIDATED STATEMENT OF INCOME
Lead-out Text:
      \t\s+52=======  =======  ======
IS Block Sparsity:
      0.6290
IS Block Num-Alfa Ratio:
      0.2131

Document Sparsity:
      0.7135
Document Num-Alfa Ratio:
      0.0486

Length of IS Block:
      59
Left Boundary of IS Block:
       1
Right Boundary of IS Block:
      78
```

## Figure 5*: Normalized Extraction of the Income Statement of IBM's 10-K of 1998

| Normalized | Raw |
|---|---|
| Revenue | Revenue: |
| Hardware segments | Hardware segments |
| Global Services segment | Global Services segment |
| Software segment | Software segment |
| Global Financing segment | Global Financing segment |
| Enterprise Investments segment/Other | Enterprise Investments segment/Other |
| | - ----------------------------------------------- |
| Cost | Total revenue |
| Hardware segments | - ----------------------------------------------- |
| Global Services segment | Cost: |
| Software segment | Hardware segments |
| Global Financing segment | Global Services segment |
| Enterprise Investments segment/Other | Software segment |
| | Global Financing segment |
| Gross profit | Enterprise Investments segment/Other |
| | - ----------------------------------------------- |
| Operating expenses | Total cost |
| Selling, general and administrative | - ----------------------------------------------- |
| Research, development and engineering | Gross profit |
| | - ----------------------------------------------- |
| Operating income | Operating expenses: |
| | Selling, general and administrative |
| Other income, principally interest | Research, development and engineering |
| | - ----------------------------------------------- |
| Interest expense | Total operating expenses |
| | - ----------------------------------------------- |
| Income before income taxes | Operating income |
| | Other income, principally interest |
| Provision for income taxes | Interest expense |
| | - ----------------------------------------------- |
| Net income | Income before income taxes |
| | Provision for income taxes |
| | - ----------------------------------------------- |
| | Net income |

* Numerical cells omitted