Information valuation for Information Lifecycle Management

Ying Chen
Storage Systems Department, IBM Almaden Research Center
650 Harry Road, San Jose CA, 95120
yingchen@us.ibm.com

Abstract

The foremost crucial step towards a fully automated Information Lifecycle Management (ILM) is to differentiate information by values in an unbiased manner and understand how values change over time. This paper presents an information valuation approach that quantifies the value of a given piece of information based on its **usage over time**. *Our case study based on several real world NFS file server traces collected from Harvard University shows that such a model is simple, effective, and tangible since it relies on measurable metrics and observable facts. It captures the changing nature of the file value throughout their lifecycles, reflects the value differences among different files, and hence allows one to compare and classify files. More importantly, through additional analysis of the model outputs one can gain new insights into files, e.g., what files are most valuable and when. We show that files in different value classes exhibit different characteristics and can be characterized by unique sets of attributes. By devising algorithms to extract such attributes automatically for different classes of files, storage systems can predict what class a file would belong to early in its lifecycle, e.g., at the creation time. The file valuation, classification, and class membership prediction can then guide a wide range of new optimizations, e.g., data placement across tiered storage and data protection.*

1. Introduction

1.1. Problem background

The world-wide electronic information is growing at more than 30% per year, reaching 5.4 Exabytes by year 2002 according to [1]. Such voluminous information poses new challenges to businesses in managing them to meet business goals, such as cost, performance, reliability, and availability, etc. Recent regulatory requirements, e.g., Sarbanes-Oxley, HIPAA, and DOD [2, 3, 4], which mandate corporations to maintain *fixed-content reference data* safely for years, impose additional complexity on information management. It is not uncommon that businesses deal with tens of terabytes of such data. Fixed content data often refers to long lasting data whose content is unchanged after some initial lifestage, but the data may still be used throughout its lifecycle, e.g., check images, contract records, X-Rays, emails, and broadcast contents.

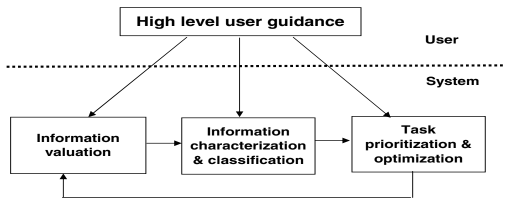
High level user guidance
User
System
Information valuation
Information characterization & classification
Task prioritization & optimization

Figure 1. The three key ILM tasks.

Recognizing that not all corporate information have the same value and values change over time, Information Lifecycle Management (ILM), a term coined by IT industry, aims at capitalizing on the information value differences throughout their lifecycles to improve system resource utilization and maximize information value automatically for fixed-content reference data. [5] described history and scope of ILM. Such automation requires storage systems to understand what data are important at what time so that right policies can be applied at the right time. Unfortunately, to date the lack of solid information valuation methods leaves ILM a wishful thinking rather than reality.

A typical ILM solution often includes tiered storage hardware and software stack that consists of storage software, middle-ware such as content management systems and databases, and other high level applications such as SAP. They work together as a whole to manage large volume of long lasting reference data throughout their lifecycles to meet business goals and regulatory compliance requirements. Often ILM-related optimization decisions, such as where a given piece of information is to be placed, are made according to the value of information. A piece of information may be a document, a file, or a database record.

In our paper, a piece of information refers to a document or a file. We will examine the effect of information granularity in the future. ILM includes three key tasks (see Figure 1): **Information valuation** lets users measure and compare the values of different pieces of information unbiasedly. With such valuation, data **characterization and classification** can then be done to guide proper **task prioritization and optimizations**. These components are iteratively refined to adapt to the changing system conditions and user requirements. High level user guidance can be useful. Yet, to automate ILM tasks, significant dependencies on humans must be avoided. This paper focuses on the first task in ILM, i.e., developing methodologies to allow storage systems to automatically calculate information value for fixed-content reference data with little or no human intervention.

Existing methods for evaluating the value of information are often crude, case-specific, and human intensive. Some value information according to vague claims, yet claims such as *"I believe this data is the most important"* are often unsubstantiated and cannot be measured. Others may value information according to the information age. Section 2 provides a more detailed summary on the prior art. We believe that for the purpose of ILM for reference data, what is needed is a generic information valuation model that is accurate, tangible, sustainable, and adaptive to a wide range of environments. Furthermore, it must be easily embedded into a fully automated ILM solution.

1.2. A big picture of our solution

We devised a *usage-over-time* based information valuation approach (UT for short), which *indirectly* infers the information value using the collected information usage statistics and some high level user guidance whenever appropriate. Under this approach, a system monitor automatically collects knowledge necessary for storage systems to measure the information value. A valuation model is derived to compute the value of a given piece of information at a given time. Our approach is based on two fundamental conjectures: *1. Information value is realized and reflected through its usage*; *2. information value changes over time*, and hence, one never refers to information value without a reference to a specific time.

One may argue that sometimes *usage* alone may not be a strong indicator of information's intrinsic values. For instance, a million-dollar transaction record is inherently significantly more important than a ten-dollar transaction even if it is not used. However, even in such situations, it is typically the case that the system activities around the million-dollar transaction record will be much more than smaller transactions. Hence, the importance of the large transactions will still be reflected through their actual usage indirectly. Furthermore, *usage* in our model may cover multiple aspects of information usage, such as usage count, the usage time, the source of usage, and the purpose of the usage, etc., rather than a single usage aspect alone. Our case study uses the usage count and recency. We plan to examine additional usage aspects in the future.

The valuation model is derived from two measurable and observable metrics: *usage* and *time*. Hence the model outputs can be substantiated. Usage statistics can be monitored and collected by systems with little human intervention. The model captures both the information value changes over time and the value differences among information. Since the usage patterns themselves are reflective to system changes, our model is automatically adaptive as shown by our real world case study.

The information value determines its *timeliness of accessibility, reliability, and availability* constraints, and drives ILM management policies and optimizations. Typical ILM policies range from data retention, placement, and migration in a tiered storage environment to data protection and recovery. The tiered storage often include expensive but fast and reliable high-end storage, cheaper but less reliable SATA-based mid-range storage, and low-cost but slow tapes. Proper ILM data placement and migration policies must determine when to place or move what data to what storage devices based on their relative values to a business throughout information lifecycle phases for maximum resource utilization. Similarly, data protection policies such as synchronous and asynchronous replication, and scheduled backup and restore must be applied intelligently for different classes of data since such policies have tradeoffs in their cost, performance, and availability guarantees.

With proper valuation, new optimizations are possible. For instance, a prioritized data recovery scheme can be developed to recover high value files first to allow the resume of core business activities, and the less valuable files in the background later on. This is especially important in disaster situations when relatively large amount of data may be lost yet only a small subset of them are critical for normal business functions at a given time period. Most existing recovery schemes do not consider such data value differences.

In addition, we make an observation that high value classes often differentiate themselves through some unique sets of attributes (that is why they are different from others), such as data types or owners. An automated algorithm can be developed to analyze the valuation outputs and extract such unique attribute sets for high value classes. We call such a process the *attribution process*. Knowing what attributes characterize high value classes further empowers the storage system with the ability to predict value class membership for a given piece of information. For instance, if it is known that files of particular types and from particular groups of users are valuable, whenever a file with those characteristics is created, the system can automatically infer its value class and apply appropriate management policies. On the contrary, if some files are known to be of little value when they are created, the system can directly put them on tapes without going through the entire tiered storage migration path. To our knowledge, no existing Hierarchical Storage Management Systems (HSM) that migrate data across tiered storage have such capabilities.

Overall, our information valuation model can be used as system tools or incorporated into system internals as part of an integrated ILM solution. Integrated system makes it easy for complete automation. The model performs information valuation periodically and automatically, such as on a quarterly or semi-annually basis. The valuation results will be used by the data characterization and classification module to extract key insights and guide ILM tasks such as data migration and protection.

The key contribution of the paper includes the following:

1. We presented a first-of-the-kind information valuation methodology that infers information value indirectly through information usage-over-time with small amount of high level user guidance.

2. We demonstrated that the model is simple and adaptive, and can be easily integrated into automated systems since it relies little on human intervention.

3. We validated our model using real world case studies and showed that the model is effective and robust.

In the rest of the paper, we discuss related work in Section 2. We present our baseline information valuation model in Section 3. We validate the model and analyze the model sensitivity in different dimensions using the real world NFS file server traces in Section 4. Section 5 discusses the baseline model limitations and presents potential future extensions. We conclude and outline future work in Section 6.

2. Related work

Existing ILM solutions often rely on traditional HSM concepts developed almost a decade ago [6] as the resort for handling data migration issues in ILM. Yet HSM is only one aspect of ILM. Even for HSM, most of these solutions migrate data through fixed migration path along the storage hierarchy based on one of the two metrics: *the last data access time* and *the data age*. Many examples of such products are available throughout the storage industry, such as IBM Tivoli Storage Manager (see http://www.ibm.com/software/tivoli/sw-bycategory/subcategory/SWJ10.html), EMC Legato (see http://www.legato.com/storage/idm). [7] discussed such a solution deployed in a real customer environment which archives files solely based on the file access *recency*. Files that have not been accessed for a while are moved from high-end storage devices to cheap storage devices. Email archiving solutions often archive data by age.

The above methods work well for strongly time-dependent data set and they work for HSM-style data migration policies alone. For other workloads, *recency* or *age* alone may not suffice. Unlike emails, bioinformatics data may not be important in their initial lifecycle phases, but they become important when they are used by scientists later on when doing DNA analysis and discovery. After that, data may be dormant for some time before a different kind of analysis is done. Our case study also indicates that some workloads may be primarily time-dependent while others are not. Furthermore, data migration and placement policies are only a subset of ILM policies. What is needed is a more systematic information valuation approach that collectively accounts for different aspects of information usage to drive a wide range of ILM policies. We present a valuation model for storage by combining both the usage access patterns as well as usage recency in one model to infer the value of data. We do not know of any prior art that combines both such factors for migration or other ILM policies. Such valuation results can also drive data placement and data protection and recovery policies. Existing ILM policies are driven by different system conditions or manual inputs, not by model-based valuation results.

Proper data classification is considered as the cornerstone of ILM [8]. The core of data classification lies in appropriate information valuation. One common data classification practice is to directly classify information by its *business criticality* (called *direct* methods). However, it is not clear how one can measure information's *business criticality* [8, 9]. Worse yet, defining *business criticality* is often not the IT organizations' expertise, and requires cross-organizational infrastructure support, which typically does not exist today. Contrasting to such *direct* methods, our approach *indirectly* measures the information value through the observed *usage-over-time*. Hence the results are more tangible and can be sustained.

[10] discussed classification methods based on the relative rankings of the applications that produce and use information. Such methods fail to distinguish data generated by the same application. Neither can they capture the information value changes over time. Hence they would only work for limited cases. Our UT model derives information value regardless of what applications created or used the information. Nevertheless, if application ranking knowledge is easily available, our model can incorporate it as well.

Recognizing the difficulties in data valuation and classification, storage vendors have also been seeking for data classification services opportunities (see http://www.emc.com/global_services/isc/im_consulting). Such services often involve ongoing expensive, lengthy, and labor-intensive analysis of customers' IT environment, data usage, and business requirements in order to make meaningful suggestions on ILM policies for a given environment. Although such lucrative service opportunities may be attractive to storage vendors at the beginning, in a long run low-cost and automatic solution is a must. Our modeling based approach aims at such a direction.

The criticality of information and its supporting IT infrastructure to the business prosperities has been a motivating factor to drive the understanding of information value in the business community [11]. [12] and [13] derived the information value based on a utility model which analyzes how the information usage leads to actual transactions and hence revenues through a set of intricate processes. The approach helps companies to understand the linkage between information and revenue. But the model is time-consuming, human-intensive, and hard to adapt to changing situations. Our UT model also infers the information value through some form of information usage. But it is uses simple and tangible metrics and automatically collected system knowledge. In the future, we will explore possibilities to automatically extract business knowledge through working with high level applications, such as workflow processes to improve our UT model accuracy.

[14] uses historical information cost, such as the cost of capturing, producing or acquiring information, to calculate information value for a business. But it fails to consider how information is used. It tends to support the creation of more data, rather than more effective use of them. Our UT model considers the information usage statistics over time and can incorporate the cost factors easily. [15, 16] highlighted methods that can be used to measure the value of knowledge and intellectual properties in order to understand their financial impact on organizations. Such business focused valuation methods are useful, yet they all require intense human interaction, special business processes, and organizational support. Hence they are often hard to implement. Our UT model avoids such problems.

Like ILM automation, the self-* storage system project in Carnegie Mellon University aims at automating storage management tasks through self-managing techniques. [17] describes how one can classify files based on automatic learning of file properties using decision tree algorithms. The work presumes the knowledge of file classes, e.g., zero-sized files or read-only files. The focus there is to learn what makes up those classes of files. Unlike self-* storage, we focus on deriving information valuation models to allow file classification by values in the first place. Once files are classified by values, techniques described in [17] may be used to identify unique attribute sets for different value classes. Such attribution methods are not the focus of this paper.

3. Information valuation modeling

To provide proper information valuation for ILM automation, the valuation model must fulfill several key requirements. It must

1. Require little or no human intervention;
2. Rely on tangible and measurable metrics;
3. Be simple and comprehensible to allow users to easily interpret the valuation outputs and gain insights;
4. Capture key trends in information values: the differences among information and value changes over time;
5. Adapt to changing environments.

We devised a *usage-over-time* based information valuation approach, based on the two conjectures as described in Section 1. A model is derived to compute the value of a piece of information for a given point of time, called *the present time*. The value computed at the present time is called the *present value*. The baseline model assumes that the past usage history serves as an indication of the importance of the information for the present time t. It indirectly infers the information value at t by factoring in two key aspects of information usage, i.e., the *recency* and the *degree* of the information usage. A piece of information is more valuable if it is used more recently and/or it is used more heavily than others. The *degree of usage* may be represented by access frequency, the length of access times, the significance of the usage to a business as captured by workflow processes in middle-ware or applications. Not all such usage factors are available in the storage layer. In this paper, we built and validate a baseline model for storage without additional high level application or user guidance using access frequency and recency factors. Hence it has its limitations as discussed in Section 5.

An effective model must combine both *recency* and *degree* of usage aspects without strong bias towards one aspect or another. Meanwhile, it must consider the tradeoffs between the two. For instance, a piece of information that was heavily used in the past may not have high value since the usage occurred long ago. Strong bias towards *recency* causes the model to assign high values to recently used data even if the usage is light. Strong bias towards the *degree* of information usage causes the model to assign high values to heavily used information regardless when the usage occurred. To eliminate such bias, we normalize the recency and degree of usage factors into a common value scale, i.e., between 0 and 1. Combining both *recency* and the *degree* of usage allows the model to capture information value changes over time and the differences among information.

Since reference data often have long life times, e.g., multiple years or months, and the majority of information are of little use some time after their births, the model needs not consider ancient usage history to compute the present information values. Yet, appropriate amount of usage history must be considered. Too short a valuation period may cause the model to fail to account for the past implications on the future, and hence the model predictions may be less accurate. We call the time duration that the model uses to compute the information values at a given time t *the valuation period*. It often starts from some time before t. The valuation computed at different times for a given piece of information will reveal information value changes over time. Our case study confirms such trends.

In practice, a valid valuation period typically should be at least a few months for long-lived reference information. This is because prominent information value changes often can only be observed in relatively long time intervals for such data. For instance, check images are often actively used in the first 30 days of their lifetimes, and are rarely or never used after 90 days. Too short a valuation period will not allow the model to capture such information usage trends. An effective valuation period can be determined by repeating the information valuation with increasing valuation period values starting from some small number of months and checking if the information valuation outputs change significantly. The valuation period is selected once the valuation outputs no longer change significantly. This is because increasing valuation period means inclusion of older usage history information. When the valuation period is long enough, additional old usage history will not affect the overall valuation outputs significantly.

The *recency* and *degree* of usage are incorporated into the model through dividing the valuation period into fixed-length *lifestages*. We assign different *recency weights* to different lifestages and track the usage statistics for each lifestage individually. The more recent lifestages are assigned with higher weights and hence the usage statistics in those lifestages will contribute more to the final information value. All usage instances occurred in a single lifestage have the same weight. The length of a lifestage affects the degree to which the model is biased towards *recency* or the *degree* of usage. Coarse lifestages reduce the effect of recency. When the length of lifestage is equal to the valuation period, the model essentially ignores the usage recency effects completely and determines the information value solely by the degree of the information usage. Short lifestages make it hard to determine appropriate recency weight assignments. In practice, given a valuation period, a small number of lifestages is sufficient for the model to generate reasonably accurate valuation outputs. Hence the length of the lifestage may be typically on the order of days or months. Section 4.3 analyzes the model sensitivity to the length of lifestage using the real world NFS file server traces. Other alternatives for selecting lifestage lengths are also possible. However, the key is to assign appropriate weights to different lifestages. Fixed-length lifestages are simple and effective as our case study indicates.

The overall valuation model is defined as follows:

$$\begin{aligned}
V_t(d) &= \sum_{i=1}^{N_t} (w(i) \times f(U_i(d))),\ 0 <= f(U_i(d)) <= 1, \\
w(i) &= \frac{(\frac{1}{x})^i}{\sum_{j=1}^{N_t} (\frac{1}{x})^j},\ \sum_{i=1}^{N_t} w(i) = 1,\ x >= 1 \\
vp &= [t - (N_t \times s), t],\ N_t = \frac{vp}{s}. \qquad (1)
\end{aligned}$$

Here $V_t(d)$ is the value of the piece of information, d, at time t. The valuation period is denoted by vp, and its duration is $[t-(N_t \times s), t]$. s is the length of each lifestage. N_t is the number of lifestages. $f(U_i(d))$ represents the normalized usage of information d in its lifestage i. Its value is between 0 and 1. We show an example of normalization function using our case study later. $w(i)$ is the normalized recency weight for lifestage i. The sum of the weights is 1. The weights are assigned using a weight function as shown above. Smaller i represents more recent lifestage. Such a weight function ensures higher weights for recent lifestages.

Given the same N_t, the larger the x is, the steeper the weight distribution is as shown in Figure 3. Similarly, given a fixed x, the larger the N_t is, the steeper the weight distribution is. In general, significantly flat or steep weight distributions should be avoided. Flat weight distribution essentially ignores the effect of the usage recency while the steep distribution considers primarily only the most recent usage and ignore all the past implications. We examine the model sensitivity to different weight distributions in Section 4.3. The key valuation model parameters are the valuation period vp, the lifestage length s, the degree of usage in each lifestage $f(U_i(d))$, and the recency weights $w(i)$.

The above model can be further normalized to some common *value scale* $[ls, us]$, such as between 1 and 10, for ease of use as follows:

$$V_t(d) = (\sum_{i=1}^{N_t} (w(i) \times f(U_i(d)))) \times us + ls. \qquad (2)$$

With a common value scale users can understand the information value distributions and develop good sense of what the different value distributions would mean over time.

Note that with our usage-over-time based approach, the model outputs will match the true information value only if the usage indeed reflects the value of information. Otherwise, there is a mismatch between the model-derived value and the true information value. When the mismatches occur, besides examining the model itself and correcting it if necessary, one should also examine other factors such as system internals and users' information usage patterns. It is possible that improper use of systems, such as wrong configurations, can lead to wrong usage patterns, and hence wrong valuation results. In such cases, correcting the wrong configuration is probably better than the model.

In particular, there are two model output scenarios when we compare them to the true information value: First, the model outputs match with the true information values well, and they can provide users with an indication on whether files have been properly differentiated. For large volume of reference data, some task prioritization and data classification are essential for the health of the system. The second scenario is that the model outputs do not match the true information value. Such valuation outputs may imply that either users have not used the system well, e.g., all information are used equally without prioritization, or the system internals have drawbacks that obscured the actual data differences. System internals may obscure the actual data differences if they use improper data structures, say. For instance, the system may store the important data together with unimportant data in the same file. Whenever the important data is retrieved, other unimportant data in the same file may also have to be retrieved. In such cases, the system internal obscured the actual data difference, and the model may wrongly consider all data to have the same value since they have the same usage patterns, although the reality is not so. It is also possible that the system is always busy processing unimportant data due to internal drawbacks or wrong configurations. In such cases, the model outputs serve as an alarm to users to examine systems more closely. Indirectly, the model results indicate how well the information resource is aligned with the business goals. The more agreeing the model-derived information value is to the user-perceived information value, the more the IT resources are aligned to the business objectives.

4. Case study

In this section, we validate and analyze our model through a real world case study. We first use our model to evaluate files in three different NFS file servers in Harvard University using 3-month traces. Then we validate and analyze the model outputs. The differences among these servers allow us to examine the model generality and adaptivity. We show that the model is effective for all three cases. Although the studied workloads are limited to a university environment currently, our modeling approach does not depend on any workload-specific knowledge, hence we believe the methodology is applicable to other workloads as well. We plan to validate with other workloads in the future. The main goals of our case study include the follows:

1. Validate if the model can capture the changes of file value over time and the value differences among files.
2. Validate if the valuation outputs reflect real file values.
3. Analyze the model sensitivity and robustness to different parameter value changes.

4.1. Trace characteristics

Validating our model requires relatively long traces for large data set. We selected NFS file server traces from Harvard University for such reasons. These file servers contain both production files that are actively modified and used and a large fraction of reference files. We processed the traces to contain the reference files as described below. Such file server environments are representative to many of today's customer environments. Often a small fraction of highly valuable files are cluttered with all other files in a single file server, or a cluster of file servers. Over time, the voluminous data slows down the file server, causes backup/restore inefficiency, and hinders the overall system productivity. This is exactly why proper information valuation is needed.

The three real workload traces are EECS, CAMPUS, and CAMPUS2. EECS is a trace of research workload from the computer science department. CAMPUS contains the workload from the main campus general purpose servers. This trace is dominated by emails. CAMPUS2 is similar to CAMPUS, although it is less busy. All three traces were gathered from 9/1/2001 to 11/30/2001. The trace sizes for EECS, CAMPUS, and CAMPUS2 are 9.5GB, 48GB, and 32GB respectively. More information on the traces are available from [18, 19, 20].

In our evaluation, we are mainly interested in fixed-content reference files with relatively long life times. We considered a file's content to be fixed after its last modification as indicated by the *mtime* of the file. That is, the modification time denotes the *birth time* of a fixed-content file. In practice, if files are still being actively modified, they often imply that they have not reached their final forms. If they are overwritten, the old contents are essentially deleted. We consider the files as new files. A trace processor is written to filter out all file operations that happened before the fixed-content file birth times, as well as files that typically do not belong to the fixed-content reference data set, e.g., short-lived files, cached html files, trash files, etc. We used file access counts to represent *usage* in our model. An access is either a *getattr* or a *read* on a file. Other operations are not counted since they do not examine the file content or their metadata. We normalized sequential reads of a file into one file read request. The final processed trace contains fixed-content reference files and their access operations. Currently we do not distinguish reads from getattrs in our valuation.

Table 1 shows the key characteristics of the processed traces. The number of files shown in Table 1 is the count of the files appeared in the processed traces. The original traces did not contain the baseline file system size information. So the file counts shown in Table 1 may be significantly smaller than the actual file set sizes. Even so more than 88% of the the files had zero file access over the 3-month trace period. These files were looked up but were never read. This percentage would have been larger should we use the entire file population. Less than 1% of the files captures more than 69% of all file accesses. This suggests that the set of files that are relevant to users at any given time may typically be a small fraction of the total data set.

	EECS	**CAMPUS**	**CAMPUS2**
# of files	2882500	969750	753833
0-access files	88%	92%	92%
high-access files	1%	0.28%	0.23%
% of accesses for high-access files	70%	69%	73%

Table 1. NFS trace characteristics.

Age (days)	**EECS**	**CAMPUS**	**CAMPUS2**
$(0, 30]$	6%	34%	32%
$(30, 90]$	23%	60%	62%
$(90, 360]$	25%	4%	3%
$(360, \infty]$	46%	2%	3%

Table 2. NFS file age distributions. $(x, y]$ indicates an age range that is larger than x and smaller than or equal to y.

Table 2 showed the file age distributions for the three file servers. 46% of the EECS files have lived more than 360 days, while most CAMPUS and CAMPUS2 files are short-lived. Only 6% of the files on CAMPUS and CAMPUS2 live longer than 90 days. This is probably mainly due to the fact that the disk space allocation on CAMPUS and CAMPUS2 file servers is more stringent than on EECS. Students often have to delete old email folders to make room for new ones. It also indicates that emails are strongly time-dependent. Hence their usage heavily depends on their ages, so are their values as shown in Section 4.2.

4.2. Model validation

To value files in the NFS file servers, we selected the following model parameter values as defaults after multiple iterations of robustness analysis. The model always generates values within the value scale $[1, 10]$.

$$\begin{aligned}
N_t &= 5,\, s = 12 \text{ days},\, vp = [t - 60 \text{ days}, t] \\
ls &= 1,\, us = 10, \text{value scale } = [1, 10] \\
U_i(d) &= \text{\# of getattrs} + \text{\# of reads} \\
w(i) &= \frac{(\frac{1}{2})^i}{\sum_{j=1}^{5} (\frac{1}{2})^j} \\
f(U_i(d)) &= \begin{cases} 1, & U_i(d) > sf; \\ \frac{U_i(d)}{sf} & \text{otherwise.} \end{cases} \quad sf = 10 \qquad (3)
\end{aligned}$$

The usage normalization function $f(U_i(d))$ uses a file access count *scaling factor*, sf, in formula 3, to normalize the file access counts to values between 0 and 1. Selecting sf must be done with care since sometimes there may be a few *outlier* files that have much higher access counts than others. If sf is too high, only those outliers may be assigned with high values, while all others are assigned with low values even if there are still significant differences among the remaining files in reality. Too small an sf will cause the model to generate high values for most of the files. In general, improperly selected sf will cause model to fail to reveal the file value differences. In our case study, we filtered out a small fraction of outliers in the file set and set the sf value to be the maximum access count of the remaining files. We study the effect of sf on our model in Section 4.3.

File value changes over time We first validated if our model is able to capture the file value changes over time. To do so, we first categorized files based on three different file access patterns: *bursty*, *periodic*, and *constant*. We consider a file a *bursty* file if all file accesses happen only in a short period of time, i.e., 30 days. Emails typically exhibit bursty patterns. In our traces, more than 80% of the files that have none-zero accesses exhibit bursty behavior. A file is *periodic* if file accesses happen in some regular interval. A file has a *constant* access pattern if it is constantly accessed throughout the valuation period. We found that files such as login shell configuration files are constantly accessed, such as on a daily basis, since students often logon to the system daily. Files with periodic and constant access patterns to have small value changes over time. Bursty files should demonstrate relatively significant value changes. Our results confirmed this as shown in Figure 2.

We randomly selected 3 files from each access pattern category and computed the file values on different dates using a 30-day valuation period and defaults for all other parameters. Clearly bursty files exhibit significant value changes over time, while the constant and periodic files showed small value changes. CAMPUS and CAMPUS2 files are similar. We do not present them here. Among those three bursty files, bursty1 had high value for about one month before its value dropped. The other two bursty

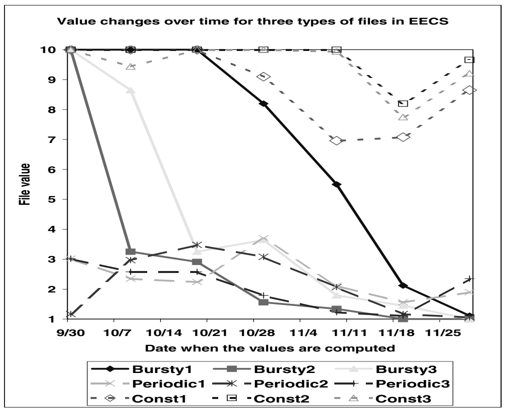

Figure 2. File value changes over time. Each line represents one file. Bursty files have significant values changes, while others have small changes. *Const* **is short for constant.**

files had much shorter bursty periods and hence their values fell much earlier. We also found that the bursts often happen at the beginning of file lifecycles (more than 80% of bursty files' bursts occured at the beginning of their lifecycles in CAMPUS and CAMPUS2). This suggests that it may be appropriate to move such files to cheaper storage media after their bursty periods. As indicated in Figure 2, for EECS the periodic files all have relatively low values since the files are only sporadically accessed. The constant files tend to have high values since they not only are accessed constantly, the access counts are also relatively high.

File value differences One key goal of information valuation is to be able to differentiate files by their values. Our traces showed clear differences among files as indicated by Table 1. Table 3 shows the valuation results by dividing files into four value classes based on our model outputs. Since the majority of the files had no accesses at all during the two-month valuation period, it might have seemed sensible to simply divide the files into two value classes, one containing all non-zero access files, and the other containing the remaining files. However, as Table 1 indicated, a large fraction of the accesses were from a small fraction of highly popular files, it would still be useful to separate out those highly popular files from others. This is why we divided files into 4 value classes rather than 2.

Value class whose value is 1 contains files with zero accesses in the 60-day valuation period. We computed the file values as of 10/30/2001 using the default model parameter values. Across all traces, more than 73% of files had zero accesses. Less than 2% of files have values larger than 5, yet they account for more than 62% of the total accesses. The

Value	EECS	CAMPUS	CAMPUS2
1	73%(0)	91%(0)	92%(0)
$(1,3]$	23%(21%)	6%(24%)	6%(26%)
$(3,5]$	2%(12%)	1%(8%)	1%(13%)
$(5,10]$	2%(67%)	1%(68%)	1%(62%)

Table 3. Value distributions for NFS files. $(x, y]$ **indicates the value range for each value class. The percentages outside of the parenthesis represent the percentages of total file counts. The percentages inside the parenthesis are the fractions of total access counts.**

value class $(1, 3]$ is the second largest class because there are significantly more files in that class, although each file had only a few accesses. Clearly the valuation results revealed the file value differences and key trends in the data set.

File valuation output validity Since today users often do not have a clear understanding of information value, we cannot directly validate the model outputs against some known true information values. However, validation can be done conceptually and indirectly. We hypothesize that different value classes must differentiate themselves through unique sets of attributes, such as size, name, type, and age, etc. If we examine the valuation outputs and reason about what inherent file attributes make some files more valuable than others and why, we can indirectly validate if the model outputs reflect the real file values. We have devised an attribution algorithm to extract unique attributes for different value classes. We do not present such algorithms here since it is a significant topic all by itself. For reference data, since typically only a small fraction of data belong to high value classes, rather than characterizing all value classes, we focus on the characterization of high value classes.

We extracted the attribute sets for value class $(5, 10]$. We found that there are several common and unique attributes that characterize this value class. For EECS, 72% of the files in this value class have constant access patterns, while other value classes are dominant by periodic and bursty files. Many constant files also have relatively long file ages: 37% of the files have longer than 360-day ages. More than 36% of the files are emails and 33% html files. Most of the html files in this value class had relatively long lives. We also found that many files from this high value class belong to specific users and/or groups. For privacy reasons, we do not disclose the actual user IDs and group IDs in this paper. We found that files from two particular users and four groups accounted for 15% and 50% of high value files respectively, and their files are often of particular types, such as *gif, pdf,* and others that have the same anonymized file extensions.

Not all of the above attributes are unique to the high value class. For instance, some html and mail files also have low values. If we simply classify all html and mail files as high value files, the high value class will include some *false-positives*, i.e., files whose values are low but were wrongly classified as high value files. One can reduce false-positives by examining combinations of attributes. For instance, the high value html files all exhibit constant access patterns. This is almost never the case for low value html files. The combination of file type and access pattern attributes can more accurately characterize files in the high value class. Knowing such attributes may empower system to predict file value class membership. For instance, if a new file that matches one of the attribute combinations that is used to characterize the high value class, it is very likely that the file belongs to a high value class.

In summary, this high value class from EECS file server indeed can be characterized through a set of common and unique attributes. The high value class in CAMPUS on the other hand is dominant by mail files with ages less than 90 days since the main workload on this server is email. Contrasting to EECS, the CAMPUS files are more bursty. Hence most of the files have high values in their earlier ages and the values drop significantly as they grow older. Such analysis revealed interesting insights into the file sets. Due to the unfamiliarity of the file server and their user environments, we cannot explain all observations. Yet, we are confident that our valuation is valuable and insightful for future optimizations.

4.3. Model sensitivity analysis

We analyze the model sensitivity to changes in recency weight distributions $w(i)$, the lifestage length s, and the scaling factor sf.

Recency weight sensitivity Recency weight distribution determines how strong the past serves as an indication for the future. We evaluated three different weight distributions as shown in Figure 3 with different degree of emphasis on the past by changing x from 1.2 to 3. Figure 4 showed how such weight distribution changes affect the file valuation output distribution for EECS. CAMPUS and CAMPUS2 are similar. We show how the value class distribution changed by presenting the percentage of the total accesses in each value class for all four value classes except for the value class 1 that contains zero-access files only.

For EECS files, there is a slight decrease as we changed x from 3 to 1.2 for value class $(5, 10]$. For value class $(3, 5]$, there is a slight increase. This is because for value class $(5, 10]$, some bursty files' values are lowered as we

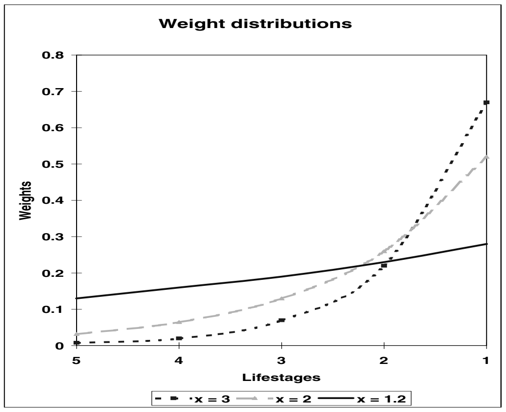

Figure 3. Three different recency weight distributions with different emphasis on the past by varying values for x. $x = 2$ is the default. $x = 1.2$ distinguishes little between the past and the present. $x = 3$ weights more recent past more heavily than others.

decreased the weights for the recent past ($x = 1.2$). For value class $(1, 3]$, some *constant* files gained values as we assign more weights to the past ($x = 1.2$), and their values fall into value class $(3, 5]$. Overall, the flat weight distribution ($x = 1.2$) made it somewhat harder to distinguish files as their value differences are smaller. We indicated earlier that too flat or steep weight distributions create strong bias towards either the *recency* or *degree* of usage aspect, and should be avoided. When the weight distributions are within reasonable range ($x = 2$ and 3), the model outputs do not change significantly. Hence it is robust to weight changes.

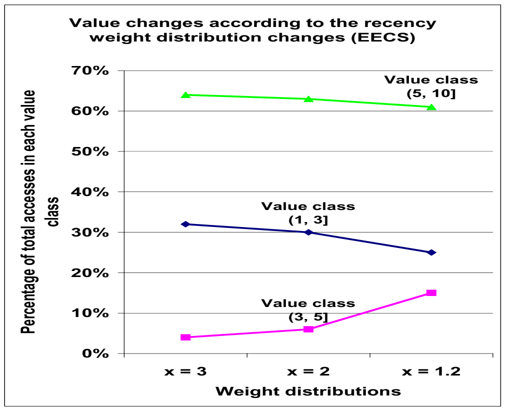

Figure 4. The effect of recency weight distribution variation on file value changes for EECS.

Lifestage length sensitivity To examine the effect of the length of lifestages on our model, we varied s from 4 to 16 days with a fixed 60-day valuation period. We are interested in the model robustness to the changes in s. The more robust it is, the easier it is to be used in practice. We used the default recency weight function for all cases. However, there is a dependent effect since the weight assignments also depends on $N_t = \frac{vp}{s}$. Since we used a fixed 60-day valuation period, the recency weights change as we vary the value for s. Such dependency is not easily removable, so our results reflect such dependent effects. An alternative is to change the length of the valuation period, which introduces a different kind of dependent effects. In our study, we tried to minimize the dependencies as much as possible.

Figure 5 and 6 showed how file value classes changed as we varied s from 4 to 12 days for EECS and CAMPUS files. For EECS, the number of files that belong to the highest value class increased significantly as we increased s from 4 to 16 days, so did the percentage of the total accesses. This is because of the relatively large fraction of constant files in the higher value classes for EECS. When we increased the value for s, we naturally increased average recency weights for all files throughout the 60-day valuation period. For constant files, since they were accessed throughout the valuation period, their values increased more dramatically than bursty files. For bursty files, higher weights for some lifestages may not affect the file values much if there are no or little accesses at all in the those lifestages. As we decrease the value for s, recency weights decrease sharply as we move into the past. Hence the past usage will contribute little to the final file value. Both EECS and CAMPUS results indicate that small s made it harder for our model to distinguish files.

For CAMPUS, small s did not affect the model output distribution as significantly as EECS. This is because CAMPUS is dominated by bursty files and their bursts often occur at the beginning of their lifecycles. As long as the model captures the usage statistics in the most recent lifestages, the model can predict information values reasonably well. Longer lifestages would not affect the valuation outputs much if there are no accesses shortly after the initial file lifestages anyway. However, we did see some valuation output differences as we increase s from 4 to 16 days. After analyzing the traces, we realize that the typical bursty periods for CAMPUS files are between 6 to 14 days. After that, the accesses become infrequent and eventually become zero. If we use too small an s, such as 4, the model would not capture the entire bursty period before the recency weight dropped significantly. In essence, the model ignored most of the history except for the most recent four days. As a result, as shown in Figure 6, most of the files have relatively small values when $s = 4$. When we increased s to 12 or 16 days, each lifestage covered much of the bursty periods, and hence the model can differentiate files with much more clarity.

Clearly for file sets with a large fraction of bursty files, we must select sufficiently long lifestage to cover the bursty file usage changes. Although the length of lifestage affect the model effectiveness, as long as s is within reasonable range (8 and 16 in our case study), the model is reasonably robust to the lifestage length changes.

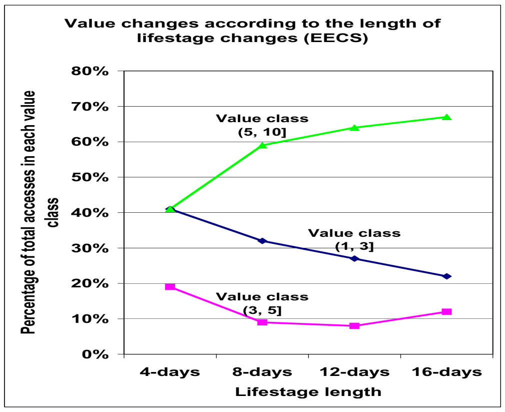

Figure 5. The effect of lifestage variation on file value changes for EECS.

Scaling factor sensitivity Different values for scaling factor sf may affect the ability for our model to differentiate files by values. We validate such effects by varying sf from 10 to 500. sf are selected based on the file access count ranges from the traces. Figure 7 showed the results for EECS. Others are similar. As we increased sf more files fall into lower value classes. The average file values decreased significantly across all three traces. For EECS, only about 400 files had file values larger than 5 when $sf = 500$, although those files still captured a large fraction of the total file accesses. Most of the remaining files fall into small value class $(1, 3]$ as indicated by the graph. Clearly, the distinction between the high value classes and others will diminish if we continue to increase sf. Note also that the reference files on these file server typically do not have a large number of accesses. Most of the files have only sporadic accesses from time to time. This is why significantly high value for sf will weaken the ability for the model to differentiate files.

In summary, our real world case study showed that the model is able to capture the key trends in information values. By examining the model outputs and extracting unique file attribute sets for high value classes, users can gain interesting insights. Our model sensitivity study showed that the model is robust as long as the parameter values fall within

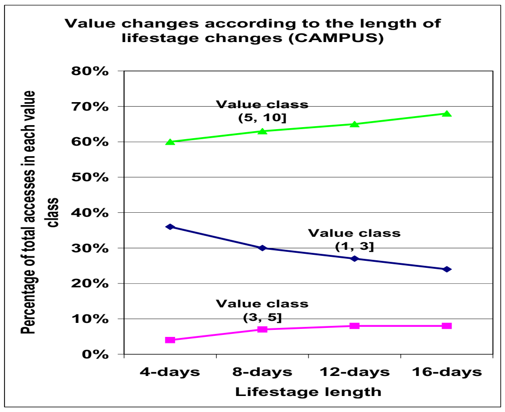

Figure 6. The effect of lifestage variation on file value changes for CAMPUS.

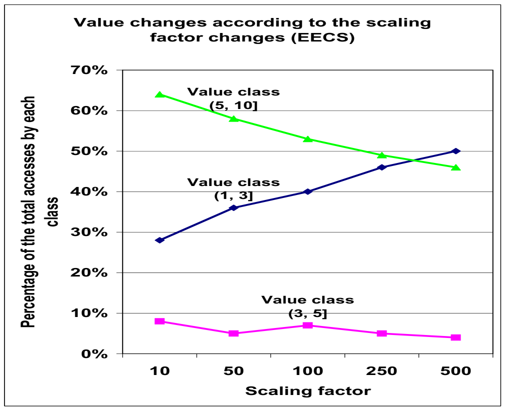

Figure 7. The effect of scaling factor variation on file value changes for EECS.

reasonable range.

5. Discussion

The baseline model for storage systems has limitations and can be improved. We discuss them below.

Incorporating future knowledge The baseline UT model uses the past usage patterns to predict the information value for the future. If users have relatively clear knowledge about the data usage for some future time, such knowledge can be incorporated through high level user guidance. Incorporating such *expected usage* can improve the model prediction. Such an extension can be done by lengthening the valuation period to include some future time and the *expected usage* in that time frame into our model. We assign lower weights for the future lifestages since future always holds uncertainty, similar to assigning weights to the past. In general, the weight distribution should be bimodal with the highest weight assigned to the present lifestage.

Incorporating high level application and user knowledge Our baseline model only considered the information collected by storage. ILM solutions typically consists of a stack of system components, e.g., storage systems, middleware, and other high level applications. Such applications often have additional knowledge about how information are used outside of storage. Similarly, users may provide their high level guidance with appropriate API support. For instance, content management systems have knowledge on how documents are linked with business workflow processes, such as analyst reporting, insurance claims, etc. By working with such middle-ware and/or high level applications, additional business knowledge can be collected and used for valuation, classification, and optimizations. For instance, the business workflow event of *report rejection* may provide implications on the value of the report.

Incorporating cost factors Our model can easily incorporate the *cost* of information production, reproduction, and acquisition as shown below. Such cost factors become important when we consider the risks of data loss. With all other usage factors being equal, easily reproducible information may be deemed as less valuable than hard-to-produce information. Here, $C(d)$ is the normalized cost factor for information d.

$$V_t(d) = C(d) \times (\sum_{i=1}^{N_t} (w(i) \times f(U_i(d)))) \times us + ls. \quad (4)$$

6. Conclusions and future work

In this paper, we presented an information valuation model that makes use of the storage system collected information usage statistics and measurable metrics to derive information value for a given piece of information at a given time. The model combines the recency and degree of usage aspects into a single function without biasing one or another, and hence it can be used in diverse environments. The overall model satisfies the key modeling requirements: it requires little or no human intervention; it relies on tangible and measurable metrics; it is simple, robust, and easily comprehensible as our case study with three file servers from Harvard University has shown. Our validation results also confirmed that the model is able to capture key trends in the data sets as well as reflect the changing information value and file value differences. Furthermore, when coupled with attribution algorithms, the overall valuation and attribute process provide interesting insights. Finally, the model is adaptive to changing environments as it uses adaptive parameters, such as usage over time.

Information valuation is only the first step towards the long term goal of ILM automation. To fully automate all ILM tasks, technologies in valuation, data characterization and classification, and optimizations must be developed. All such technologies also need to be integrated coherently across system stack as fully automated ILM solutions. In the future, we plan to continue our work in those directions.

7. Acknowledgments

We thank colleagues in the IBM Almaden research center for their support and feedback on this work, including Windsor Hsu, Shauchi Ong, Sandeep Uttamchandani, and Honesty Young. We also thank reviewers and Professor Karsten Schwan, our shepherd, for their valuable review comments.

References

[1] P. Lyman, H. R. Varian, K. Swearingen, P. Charles, N. Good, L. L. Jordan, and J. Pal. How much information? 2003. http://www.sims.berkeley.edu/research/projects/how-much-info-2003/, October 2003.

[2] Public Law 104-191, Health insurance portability and accountability act. http://aspe.hhs.gov/adnsimp/pl104191.htm, August 1996.

[3] Public Law 107-204, Sarbanes-Oxley, An Act. http://corporate.findlaw.com/industry/corporate/docs/-pub107.204.pdf, July 2002.

[4] DOD 5012.2-STD. Design criteria standard for electronic records management software applications. Assistant Secretary of Defense for command, control, communications, and intelligence, June 2002.

[5] M. Peterson. Information Lifecycle Management: A vision for the future. http://www.snia.org/tech_activities/dmf/SRC-Profile_ILM_Vision_3-29-04.pdf, March 2004.

[6] J. Menon and K. Treiber. Daisy: A virtual-disk Hierarchical Storage Manager. *SIGMETRICS Performance Evaluation Review*, 25(3):37–44, December 1997.

[7] C. Johnson. ILM Case Study: Complete Data Lifecycle Management Solution. http://www.snia.org/tech_activities/dmf/ILM_Solutions_Conference/2004/Agenda/Mustang, October 2004.

[8] Data Classification: The cornerstone for successful Information Lifecycle Management. http://www.adexisstorage.com/ilm/EMC/EMC_ILM_data-_classification.pdf, September 2003.

[9] P. Black. Data Protection Initiative Introduction. In *Storage Networking World, http://www.snia.org/tech_activities/dmf/dpi/DPI_Documents/-SNW_Phoenix_DPI_Intro.pdf*, Phoenix, Arizona, April 2004.

[10] E. Pierre. Introduction to ILM: A Tutorial. http://www.snia.org/tech_activities/dmf/ILM_Solutions_Conference/2004/Agenda/IntroductionILM, October 2004.

[11] P. A. Strassmann. Getting better value from information management. *Information Economics Journal*, 2003.

[12] R. Glazer. Measuring the value of information: The information-intensive organization. *IBM System Journal*, 32(1):99–110, 1993.

[13] R. Poore. Valuing information assets for security risk management. *Information systems security*, 9(9):17–23, 2000.

[14] D. Moody and P. Walsh. Measuring the value of information: An asset valuation approach. In *European Conference on Information Systems*, 1999.

[15] K. E. Sveiby. The balanced scorecard (bsc) and the intangible assets monitor – a comparison. http://www.sveiby.com/articles/BSCandIAM.html, 2001.

[16] P. A. Strassmann. The value of computers, information and knowledge. http://www.strassmann.com/pubs/cik/cik-value.shtml, January 1996.

[17] M. Mesnier, E. Thereska, G. R. Ganger, D. Ellard, and M. Seltzer. File classification in self-* storage systems. In *Proceedings of the First International Conference on Autonomic Computing (ICAC-04)*, New York, NY, May 2004.

[18] D. Ellard, J. Ledlie, P. Malkani, and M. Seltzer. Passive NFS tracing of email and research workloads. In *Proceedings of FAST'03*, pages 203–216, San Francisco, CA, March 2003.

[19] D. Ellard and M. Seltzer. New NFS tracing tools and techniques for system analysis. In *Proceedings of LISA'03*, San Diego, CA, October 2003.

[20] Harvard. htpp://www.eecs.harvard.edu/sos/traces.html.