Enterprise Computing Systems as Information Factories

K. Mani Chandy, Lu Tian and Daniel M. Zimmerman
Computer Science 256-80
California Institute of Technology
Pasadena, California 91125 USA
{mani, lutian, dmz}@cs.caltech.edu

Abstract

The analysis, and eventual approval or rejection, of new enterprise information technology (IT) initiatives often proceeds on the basis of informal estimates of return on investment. Investment in new IT initiatives includes the costs of hardware, software licenses, application development tailored to the enterprise, and maintenance. Returns are typically estimated informally in terms of cost savings or revenue increases. This paper makes the case for evaluating certain IT investments in the same way as investments in factories and other resources have been evaluated for decades. Just as industrial factories create value by transforming raw materials into finished products, some IT investments, which we call "information factories", create value by transforming raw information (events) into structured data (and possibly actions based on that data). The return on investment is estimated by the difference between the economic value of the structured data and concomitant actions (the "finished product") and that of the data available within the enterprise, from its partners and customers, and from the Internet (the "raw materials"). This paper introduces the concept of the information factory, and explores design considerations for maximizing the economic efficiency of information factories.

1. Introduction

As sensors and RFID tags become more widely deployed, and streams of data from commodity exchanges, wire services, and other sources become more widely and freely available, new application areas for enterprise computing systems are emerging. Like an automobile factory, which takes raw materials such as steel and glass as "inputs" and creates value by transforming them into cars, an enterprise computing system can take raw events generated by myriad sources as inputs and create value by transforming them into more structured information. Such structured information can result in actions, and for this reason is sometimes called "actionable" information; the more high-quality actionable information an enterprise has, the more quickly and more appropriately it can react to threats and opportunities in its extended environment.

A *stream application* that transforms raw events from data streams in this way enables an enterprise computing system to serve as an *information factory*, capable not only of detecting threats and opportunities but also—at least in some cases—of reacting to them autonomously. The economic value added to an enterprise by a industrial factory, measured in dollars per unit time, can be estimated from the rates of its economic input (the raw materials) and output (the finished products); industrial engineering and job-shop designs have focused on optimizing this means of adding economic value. An information factory takes streams of raw information—stock and commodity prices, foreign exchange rates, interest rates, information about events relevant to the enterprise (such as hurricanes), and information about events within the enterprise—as input and generates streams of processed information as output. The nature of these output streams varies widely across industries: a financial services company's system may output messages that cause stocks and commodities to be bought and sold, an airline's system may output messages that cause ticket prices for specific fare classes on various flights to be changed, and a power company's system may output messages that switch particular generators on or off.

Analysis of the ROI (return on investment) for IT initiatives is often carried out informally and results in disagreements between business executives, such as the CFO, and the IT department and its CIO. Measurement and accounting techniques for "brick and mortar" resources such as factories and real estate have been developed and improved over decades, whereas similar techniques for IT investments are relatively new. This short paper makes the case that some IT investments can be understood in brick-and-mortar terms: the benefit provided by an information factory is similar to the benefit provided by an industrial

Proceedings of the 10th IEEE International
Enterprise Distributed Object Computing Conference (EDOC'06)
0-7695-2558-X/06 $20.00 © 2006 IEEE

factory, namely the difference in value between the finished product and the raw material used to construct it.

As the fraction of the world economy built around services becomes greater, it becomes more important to understand and evaluate enterprises in terms of their information factories as well as their traditional factories. An example of the integration of traditional and information factories within enterprises is the changing role of automobile companies. Increasingly, customers buy cars (produced by traditional factories) with ongoing services such as OnStar from General Motors (produced by information factories). In the case of OnStar, an information factory receives events from cars and can provide great added value by, for example, automatically dispatching emergency services to the car's location if the car is in an accident. OnStar processes raw material—events from the customer's car—for many years after the customer buys the car, and the added value of this processing is quantified in the subscription fee the customer is willing to pay. As more customers use these services, the information factory becomes an increasingly important source of revenue.

Another common example of an information factory is "straight-through processing" (STP) of transactions on brokerage accounts. Clients buy and sell stocks and commodities; these operations produce messages with (generally) well-defined schemas. Straight-through processing has many steps: the client's ID and other security information is checked, the proposed transaction is checked against clients' rules for their portfolios, the buy/sell orders may be partitioned into multiple smaller orders, the orders are executed, resulting costs of execution are computed, and so on. As streams of events—STP requests—arrive at a brokerage house from different types of customers, the mapping of these streams to available hardware resources is important in generating the maximum profit by providing better quality of service for requests from high-revenue customers.

Information factories provide different benefits to various types of enterprises, including traditional manufacturers of goods (*e.g.*, apparel, office supplies), financial services organizations such as mutual fund companies, companies trading commodities such as electricity, and "Web 2.0" companies like Google and Yahoo. This paper is too short to discuss the benefits of information factories for each of these types of enterprises separately and in detail. Instead, we briefly discuss some of the key aspects of information factories in general and some design considerations for them.

2. Information Factory Characteristics

The following are the five critical differences between the industrial factory of the past and the information factory of the present that influence information factory design.

Response Time The value added by the information factory depends critically on response time (also called *latency*), the delay between the arrival of raw information and the production of actionable output. For example, an arbitrage opportunity may disappear in seconds, so the economic value (or *utility*) of identifying arbitrage opportunities is directly related to how quickly they are identified. The difference in utility between a response in one millisecond and a response in one hour depends on the application. In a customer service application, a response need only occur in time to deal with a customer's problem while they are on the phone; a response in one second provides as much utility as a response in one millisecond. In a supply chain application dealing with trucking logistics, a response in one minute may provide as much utility as a response within milliseconds. Thus, a careful analysis of the utility of responses as a function of the quality of the response and the response time is important.

Accuracy The actions taken by an enterprise based on the output of an information factory may be incorrect. The appropriateness or accuracy of a response has a dramatic effect on the value added by the response. One way of partitioning some of the incorrect decisions is in terms of *false positives* and *false negatives*: a false positive arises when the system detects and responds to a threat or opportunity that does not exist; a false negative arises when the system either detects a threat or opportunity too late for a useful response, or does not detect it at all. Accuracy plays a role in industrial factories too; for instance, some automobile factories produce more reliable cars than others. The critical economic differences in outputs with varying accuracy is, however, a key characteristic of the information factory.

Adaptability The information factory adapts continuously: new users and new computers are added almost daily, new network capabilities are added very frequently, and new applications are added as needs arise. Industrial factories adapt too, but an automobile factory or a refinery changes far more slowly than an information factory. This constant adaptation plays a role in the models used to design and evaluate information factories.

Compositionality Creating an information factory that uses the outputs of other information factories as raw material and produces value-added streams of information as finished products is much easier than doing the same with brick-and-mortar factories. Amazon and eBay offer Web Services that are used by third parties to create value-added services. So-called "mashups" [1] are a critical and integral part of "Web 2.0". Also, unlike traditional goods and services, the cost of duplicating information streams to feed increasing numbers of information factories is very small.

Proceedings of the 10th IEEE International
Enterprise Distributed Object Computing Conference (EDOC'06)
0-7695-2558-X/06 $20.00 © 2006 IEEE

Thus, the value of Amazon's finished products, such as the Web Services it offers, increases rapidly with each organization that exploits these services to produce even more services; the incremental cost of supporting more users is relatively small.

Personalization Information factories provide personalized services to their customers. Compare the personalization of an automobile to that of the services offered by Yahoo. In the former case, the efficiency and success of the factory depends crucially on the product being standard and relatively depersonalized. By contrast, in the latter case the value provided by the information factory is in personalizing its services to the precise needs of each customer at any given time.

These five differences between traditional factories and information factories play an important role in understanding the return on an information factory investment.

3. Return on Investment

The creation and continued operation of an information factory has three phases, which are repeated many times over the life of the factory: (1) capacity planning, *i.e.*, investment in resources such as processors, software licenses, application development, and communication bandwidth to run the expected application load over a relatively long time horizon (such as a year); (2) mapping of tasks to resources [8]; and (3) adaptation of task schedules to optimize resource utilization [6, 10]. Capacity planning is carried out relatively infrequently, perhaps once a year. The analysis of ROI at this stage deals with the costs of the additional investments—the new hardware, licenses, and development and deployment costs. Mapping and re-mapping of stream processing tasks to resources occurs more frequently, perhaps once a month. In this case, the investment is fixed: the supply of resources is inelastic. Schedule adaptation to maximize economic utility happens continuously. Here too, the supply of resources remains unchanged. Thus, ROI is estimated differently for each of the phases. These three phases occur in traditional manufacturing plants as well as in information factories, though not necessarily at the same frequencies. Here too, we can benefit from decades of earlier work on accounting and investment in factories to understand and optimize information systems.

An information factory can operate in two modes: *normal* mode and *sense-respond* mode. Most enterprise applications—such as customer relationship management, supply chain management, enterprise resource planning, and factory automation—operate primarily in normal mode: the raw materials and finished products are well defined and are often structured with predefined schemas. Sense-response mode, by contrast, identifies threats and opportunities; this mode of operation is used for applications such as arbitrage, as well as traditional applications such as customer relationship management (CRM), supply chain management (SCM), and enterprise resource planning (ERP). Traditional applications are monitored to determine when systems reach critical states that are precursors to threats such as the inability to provide services to key customers; impending threats are sensed (detected) and, ideally, the system responds to them before they escalate.

The value of an information factory's finished product depends on whether it operates in normal or sense/response mode. The value of data produced by systems running in normal mode has been studied more extensively than that of data produced by systems running in sense-respond mode. Inaccurate data is more of a concern in sense-respond mode than in normal mode, because it may identify a false threat or opportunity. By contrast, the data sources feeding normal mode applications are used repeatedly over long periods, so problems with accuracy can be worked out over time. In our discussion of stream applications, we deal more with the sense-respond mode than with the normal mode.

The cost of raw materials is the cost of acquiring data from within the enterprise, from its customers and partners, and from sources outside the enterprise. The cost of acquiring data from outside the enterprise—from sources such as competitors' web sites and government publications—includes the cost of repeatedly polling for and extracting information from new unstructured data (natural-language text and images). Data from outside the enterprise, in addition to being more expensive than data from sources within the enterprise, is also more likely to be inaccurate because of its comparative lack of structure. The return on investment for a stream application is estimated using the benefits realized by the application operating in normal and sense-respond modes, the costs of acquiring input data, and the costs of processing the input data through the factory.

4. Stream Applications

The problem addressed by information factories operating in the sense-respond mode has three main parts: (1) specifying threats and opportunities (*critical states*) that are important to the enterprise and the actions (automated or otherwise) to be taken in response; (2) detecting these critical states; and (3) carrying out the response actions. The second of these, which we call *critical state detection*, is the most computationally intensive part of the information factory and the part where stream applications play a role; the specification of situations and responses, and the actual execution of responses, are either primarily handled by humans or are simple computations (such as placing an automated

Proceedings of the 10th IEEE International
Enterprise Distributed Object Computing Conference (EDOC'06)
0-7695-2558-X/06 $20.00 © 2006 IEEE

stock trade or triggering the shutdown process on a power generator). We therefore focus primarily on the design and implementation of stream applications.

A stream application, in general, is an application that takes one or more streams of data as inputs and generates a stream of data as output. The particular type of stream application used in an information factory for critical state detection, which we call a *detection system*, fuses—correlates, integrates and analyzes—the data that it obtains from multiple streams and generates notifications of critical states. A critical state can be defined as one in which measurement data fits a *model* of critical states or as an *anomaly*. At each point in time, the detection system infers the most likely states from the data available up to that point and generates notifications accordingly. The system continually updates its inferences as time passes and as new events arrive.

A detection system has a set of models, some of which represent normal situations (no supply chain disruptions, no arbitrage opportunities, *etc*.) and some of which represent critical states (*e.g*., an impending shortage of a critical manufacturing part). A model of an arbitrage situation deals with the evolution of multiple markets over time; a model of a supply chain disruption may deal with specific performance indicators relating to a set of suppliers that fit patterns over time and geographical locations. Models are based on the history of *global states* of the environment being monitored, where the history at time T is a function that specifies the state $s(t)$ of the system for t ranging from the point of system initiation ($t = 0$) to the current time ($t = T$). For example, in an application dealing with financial markets, the global state may include market indicators (such as the Dow Jones Industrial Average), the current prices of specific commodities, the volume of shares being traded, the time and details of the last trade executed, *etc*.

An *anomaly* is a situation that does not fit any of the models in the detection system, and can therefore not be immediately classified. When an anomaly is detected, further analyses must be carried out by humans to determine whether or not the anomaly indicates a critical state. After an anomalous situation is analyzed, a model may be added to the detection system to handle future occurrences of the situation.

4.1. Graph Representation

A detection system must carry out many simultaneous computations on the same set of event information in order to evaluate the applicability (or lack thereof) of its various models to the global state. The computations that a detection system must perform on the event information vary widely depending on the models; in general, however, the computations can be represented by a directed acyclic graph in which the nodes represent computational steps and the directed edges represent data flowing between steps. A node may represent the evaluation of natural language text messages (such as newsletters sent by competing companies to their customers) to determine whether they are relevant to a critical situation, or the evaluation of images (such as photographs of a production line) to determine whether they suggest situations that merit further investigation. Nodes may also represent incremental statistical analyses such as regressions and change-detection algorithms, or merely simple Boolean operations such as conjunction. Some nodes may execute multiple models to determine which one best explains the current data. Thus, the computational granularity of nodes varies widely.

The *sources* (nodes without predecessors) of the graph represent event sources, and the *sinks* (nodes without successors) represent notifications. For example, one source node may represent a stream of images from a video camera while another source node represents a sequence of measurements sent by a temperature sensor. Similarly, one sink node may represent a notification that alerts security officials in a specified location while another sink node represents a notification that shuts down a production line.

4.2. Incremental Computation with Asynchronous Noisy Data

Detection systems continuously process information from multiple streams as data arrives. In general, the state of a given sensor or information source changes relatively slowly, so that at each instant in time the global state is either unchanged or is changed by only a small amount relative to the state at the previous instant. Therefore, efficient algorithms communicate and compute incrementally based on the changes between the previous state and the present state.

Data may be noisy. Also, there may be a delay between the instant at which a measurement is made by a sensor and the instant at which a message containing that measurement data is sent from a source node. While a source node *represents* an event source, it is (usually) not the actual event source, and must receive communications from outside the detection system. Therefore errors—false positives and false negatives—are possible. The challenge is to design systems that have tolerable error rates, process streams of data from all the data sources at the rates at which they are generated, and respond with adequate response times.

There are various ways to design detection systems to perform these incremental computations, but they can be classified into two types: process-on-arrival and snapshot-based. Likewise, there are various ways to design signaling schemes—algorithms that determine when sensors and other data sources send messages. The tradeoffs in the design of signaling schemes have been investigated previously

Proceedings of the 10th IEEE International
Enterprise Distributed Object Computing Conference (EDOC'06)
0-7695-2558-X/06 $20.00 © 2006 IEEE

[2]; for the purposes of this discussion, we assume the use of a *predicate-driven* (or *model-driven*) signaling scheme, where computational elements send messages only when their outputs deviate from pre-specified models.

4.3. On-Arrival *vs.* Snapshot Processing

One possible architecture for the detection system is a distributed system—either deployed on an enterprise computing system or "emulated" within a single shared-memory machine—with exactly the same structure as that of the computation graph: it has one processor (or process) for each node, and the directed edges of the graph represent message channels between nodes.

In a process-on-arrival system, each process remains idle until one of its input channels becomes non-empty, at which point it executes a step by removing a message from one of its input channels and carrying out the incremental computation associated with that message. This execution may result in the generation of messages to other processes.

Each message generated by a data source is assigned a timestamp. It is assumed that the clocks at the data sources are synchronized in a lightweight way, such that the amount of clock drift can be bounded. For snapshot processing, time is partitioned into a sequence of contiguous intervals, and it is assumed that all messages having a timestamp within the same interval belong to the same global state. All messages with timestamps in the interval $[k \times D, (k+1) \times D)$, are treated as though they reflect the kth global state, and are processed as an atomic unit. If multiple messages on the same channel have timestamps that place them in the kth global state, all but the last one are discarded.

The value of D is determined by the rates of change of sensor values. A D that is too large may encompass several changes in the measurements made by one sensor; since only the last of these measurements will be used to represent the entire interval, this can result in less accurate estimates of global states. A D that is too small may result in large numbers of process activations and a consequent performance penalty. We call each time interval in a snapshot processing computation a *computation phase* (or simply a *phase*).

We have devised multiple algorithms to schedule the execution of nodes in a computation graph for snapshot processing. For space reasons, we do not describe these algorithms here; a complete description of one of them, including some simulation results, can be found in a previous paper [9]. Descriptions of the others, and further simulation results, are forthcoming.

4.4. Resource Allocation for Stream Applications

An important consideration for information factories is resource allocation: how to best utilize the computational resources available to the enterprise to execute the stream applications that comprise information factories. In the model we have described, each stream application consists of a graph of multiple interacting and *repeatedly* executed processing units. Conventional resource allocation methods are not suitable for stream applications, for the following reasons:

Frequency of Execution Each repetition of the execution is triggered by the arrival of new inputs, the frequency of which can have a fixed rate, follow a specified distribution, or be completely random. Therefore, methods that associate each task with a fixed deadline are not suitable, because the exact time at which each task execution can start is not known *a priori*.

Quality of Service Requirements Stream applications have varying quality of service requirements; some require processing and analysis to be done on-the-fly in order to enable real-time responses, with little tolerance to delay. They also have elastic deadlines, where the value realized depends on how quickly the inputs are processed. Thus, methods that use a priority number to indicate a task's importance/value are also not suitable.

Optimization for Economic Value Traditional resource allocation algorithms optimize metrics such as makespan (the total time required to execute all tasks) or the number of tasks completed before a specific deadline. However, our objective is to maximize the net economic value of the tasks in order to maximize the return on investment for information factories. Thus, the existing methods that optimize traditional metrics are not suitable for our optimization.

We have devised resource allocation heuristics for stream applications that use microeconomic principles to maximize the utility of the stream applications (and therefore to maximize ROI for information factories). For space reasons, we do not describe these heuristics here; their complete descriptions, proofs of their correctness and efficiency, and simulation results are available elsewhere [7].

5. Summary and Future Directions

The central point of this paper is that return on investment for certain information technology applications can be analyzed in the same terms used for traditional brick-and-mortar factories; the return is evaluated as the difference

Proceedings of the 10th IEEE International
Enterprise Distributed Object Computing Conference (EDOC'06)
0-7695-2558-X/06 $20.00 © 2006 IEEE

between the economic value of the finished products and the economic value of the incoming raw materials. The finished product in the case of the information factory is actionable structured information, and the raw material is unstructured, partially structured or structured information on the Internet and within enterprise databases and applications.

We have identified five characteristics that differentiate the information factory from traditional factories that process raw materials: the variation of utility with response time, the variation of utility with accuracy, adaptability, compositionality, and personalization. Theories and tools for evaluating and optimizing traditional factories can be adapted for information factories, and the process of adaptation must take into account these key differences between information factories and traditional factories.

Snapshot processing deals with accuracy; the raw data arriving at the factory measures various aspects of the enterprise, and this raw data is pieced together to form a sequence of global snapshots over time—an approximation to the overall trajectory of the enterprise. This trajectory is then analyzed to detect, and enable responses to, threats and opportunities. We have analyzed several algorithms for snapshot processing and obtained definitive performance results.

Resource allocation enables an enterprise computing system to be used in a way that optimizes the economic production of the information factory. We have evaluated market mechanisms for mapping stream applications to resources—much like mapping tasks in manufacturing plants to work stations—but with one critical difference: different utilities for different response times and different types of information.

A great deal of research remains to be done. The economic analyses of the accuracy of information produced by the factory should be studied in depth. Just as certain brands of cars gain the reputation of being more reliable than others, so too can some information streams gain reputations for reliability. Information considered to be dependable is used in decision making, while unreliable information is ignored. The economic consequences of accuracy and timeliness are self-evident with information on the World Wide Web, but careful analyses of the marketplace for information in terms of information factories have yet to be carried out.

Another area of continuing research [5] is in making the information factory adapt to changing loads and changing requirements by dynamically adjusting priorities for its component tasks. The response time that matters most to customers is the overall time from the arrival of raw data to the production of actionable information; in effect, the response times for intermediate steps are irrelevant. Therefore, information factories can adapt the scheduling of intermediate steps to maximize overall utility.

Much has been written about autonomic computing [4], the central nervous system of computing [3], and the real-time enterprise. The key element in these systems is the information factory, and the key question for many enterprises is how much return they can realize on an investment in information factories. This paper is an initial step toward exploring answers to this question.

Acknowledgements

The research described in this paper has been supported in part by the National Science Foundation under grant CCR-0312778, ITR: Information Infrastructures for Crisis Management, and by the Lee Center for Advanced Networking at Caltech.

References

[1] D. Butler. Mashups mix data into global service. *Nature*, 439(7072):6–7, Jan. 2006.

[2] A. Capponi and K. M. Chandy. Predicate signaling in distributed sensor networks. Technical Report CaltechCSTR:2006.002, California Institute of Technology, Jan. 2006.

[3] W. H. Gates, III. *Business @ the Speed of Thought: Using a Digital Nervous System*. Warner Books, 1999.

[4] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, Jan. 2003.

[5] A. Khorlin. Scheduling in distributed stream processing systems. Master's thesis, California Institute of Technology, 2006.

[6] A. Khorlin and K. M. Chandy. On optimal control of computational streams. In *First International Workshop on Event-Driven Architecture, Processing and Systems (EDA-PS'06)*, Sept. 2006.

[7] L. Tian. Resource allocation in streaming environments. Master's thesis, California Institute of Technology, 2006.

[8] L. Tian and K. M. Chandy. Resource allocation in streaming environments. In *7th IEEE International Conference on Grid Computing (GRID 2006)*, Sept. 2006.

[9] D. M. Zimmerman and K. M. Chandy. A parallel algorithm for correlating event streams. In *19th International Parallel & Distributed Programming Symposium (IPDPS 2005)*, Apr. 2005.

[10] D. M. Zimmerman and K. M. Chandy. Snapshot processing in streaming environments. In *7th IEEE International Conference on Grid Computing (GRID 2006)*, Sept. 2006.