INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITE DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

DISTRIBUTED DATA SOURCES FOR LIFECYCLE DESIGN

Vijitashwa Pandey [1], Deborah Thurston[1], Khushboo Kanjani[2] and Jennifer Welch[2]
[1]University of Illinois at Urbana-Champaign
[2]Texas A & M University

ABSTRACT
The rapid expansion of the cyber-infrastructure has resulted in a wealth of data that previous generations of product designers could only dream of. It is now possible to obtain real-time data at all points in the product lifecycle. But without an organized framework for gathering, analyzing and making design decisions with it, this information goes to waste. This paper presents a method for tapping into this data without being overwhelmed by it. An ideal case study is product design for cost-effective compliance with product take-back laws. A previously developed constrained optimization model indicates that data from sources throughout the product lifecycle can be used make better design decisions regarding component reuse and remanufacture which simultaneously decrease cost and increase customer satisfaction. However, computational issues previously limited the analysis to a single set of static, industry average values for model inputs. Real-world implementation requires dynamic input from a variety of widely distributed data sources, ranging from material suppliers to the customer. However, existing computer programming methods that efficiently utilize widely distributed data are limited. This paper makes advances on two fronts. First, a decision model is presented which effectively utilizes distributed data sources regarding both cost and customer preferences. Second, the system is made fault tolerant by using the existing distributed shared memory infrastructure which involves the concepts of replicated data and quorums. Case study results indicate that information from distributed sources can be efficiently acquired using multiple replicas of data and a probabilistic read and write, leading to improved customer satisfaction.

Keywords: Product take-back, distributed data, optimization, tradeoff decisions

1 INTRODUCTION
It is now possible to obtain real-time data at all points in the product life-cycle, from conceptual design, prototyping, materials acquisition, supply chain, manufacturing, assembly, inventory, point-of-purchase, consumer use and disposition. This data conveys a wealth of up to date information, including costs and customer preferences. It could potentially be used to design better products, but its sheer volume can overwhelm a designer. Two things are needed; an analytic framework for determining which data are relevant and for using the data effectively during the design process, and also a computational approach to gathering, storing, and transmitting data among distributed sources. This paper presents a solution to these problems, and employs a design for product take-back case study. The next section provides background on the case study and on the computational approach employed, followed by an illustrative example. The final section presents results and discussion.

2 BACKGROUND
This section provides a brief background on related research in product design related to take-back legislation, and introduces the design model on which this paper expands. Product take-back laws have been enacted in the European Union and Japan. In the United States, twenty four states have active or pending product take-back legislation. For example, the Waste Electrical and Electronic Equipment (WEEE) directive sets the target of recovery and reuse at 75% by weight of post-use home appliances and computer products [1]. Similarly a target of 85% recovery and 80% recycling by weight was set for end-of-life vehicles [2]. These laws aim to "internalize the externalities" to some degree by making the manufacturer and/or consumer responsible for recovery and possible reuse.

Designers1 need to consider post-use alternatives during the product design process in order to comply with take-back legislation cost-effectively. Several such models have been developed. Ferrer [3] presents a case study on personal computers which demonstrates that remanufacturing computers is a viable option. Sandborn and Murphy [4], present a model for incorporating economic and environmental issues into product design. Campbell and Hasan [5] present a model for determining the feasibility of recycling by tracking disassembly and monetary gain for recyclers. Bhamra et al., [6], demonstrate the potential benefits of component level reuse and remanufacturing.

Models discussed so far are limited as they employ industry average static data. Real life applications include dynamic data inputs from various sources. Incorporating these inputs into the decision models is the next logical step in long range product portfolio planning. However, limited computational capabilities have restricted research in this area. In this paper we propose a model to make product decisions incorporating dynamic data input from distributed sources. The model is formulated in such a way that it is fault tolerant and highly available.

The work on which this paper builds [7], [8], combines environmental impacts with cost and reliability into a long range product planning model shown in Figure 1. Starting from the upper left and moving clockwise around Figure 1, upon take-back, 88 components of one product are analyzed, and four possible decisions are made about each component for a second lifecycle. The possible decisions include using a new component, reuse, remanufacture, or recycle.

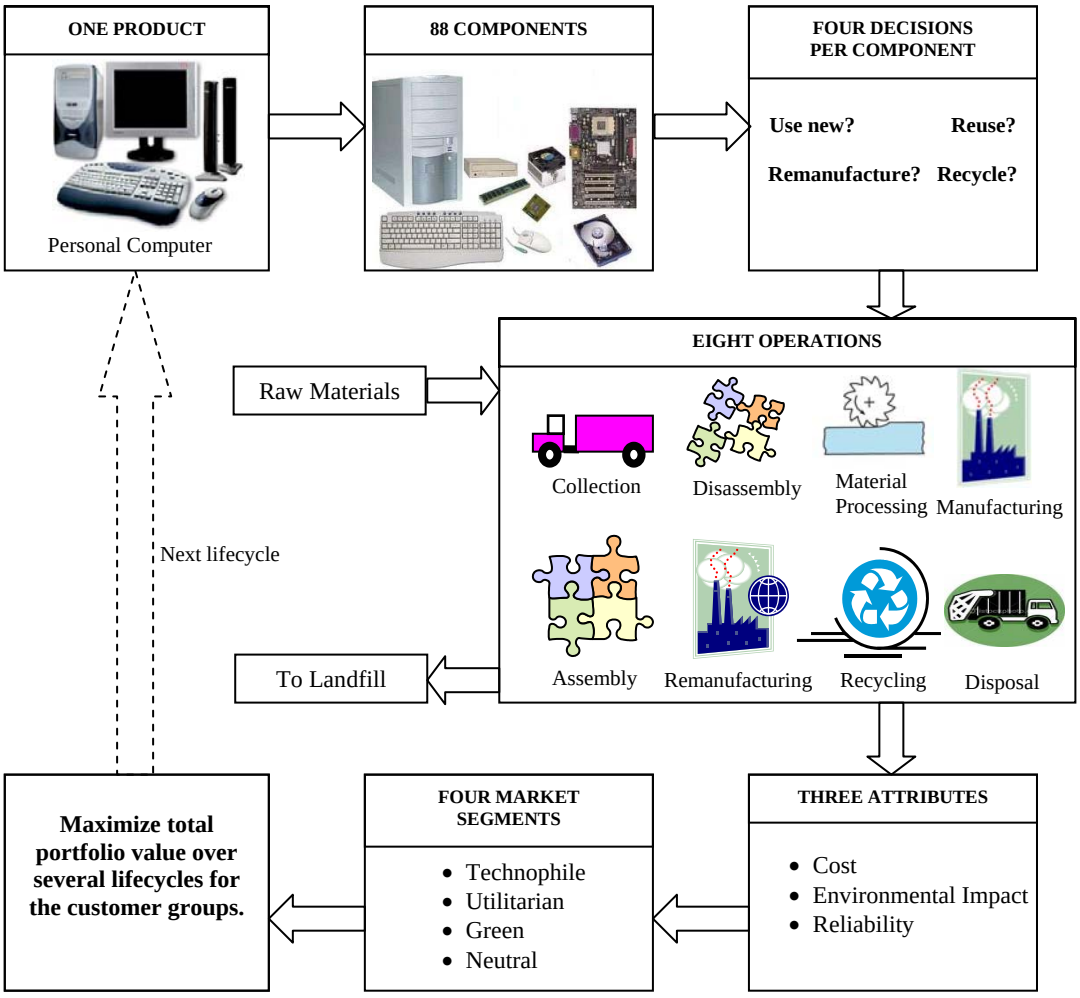

Figure 1: Design Optimization for Component Reuse, Remanufacturing or Recycling

The eight operations shown in Figure 1 are then considered for their resulting cost, environmental impact and product reliability. Four market segments are considered, so the designer can create the optimal set of products that meet the specific preferences of each segment, and simultaneously exploit

their willingness to pay for various features. Market segmentation is a powerful tool for characterizing the preference distribution of the population in general. While it would be impractical to document and cater to the preferences of individual customers, dividing them into a manageable number of segments was shown to improve total customer satisfaction. Total portfolio utility over several market segments and lifecycle is then maximized. Further work [9] indicated that adopting leasing as a business model gave the designer greater control over the take-back process, and improved the optimal solution.

3 DISTRIBUTED DATA

The data sources of interest to our model are widely distributed. They fall into two general categories. The first is data that has system-wide implications, such as updated material and manufacturing process cost parameters. The second category is customer-specific preference information. Both data types are widely distributed and change dynamically. It is important to periodically recompute the solution to the optimization problem with the latest values of the parameters.

We want the data to be available, even in case of failures in the computer network. A well-known approach (e.g., [10]) to providing highly available shared data in computer systems is to use replication: for each logical (or virtual) piece of data, there are actually multiple copies of it, stored at various locations in the network. An obvious cost of replication is the use of additional storage.

The potential benefits of replication are increased reliability, since there is more than one copy (so if one copy becomes unavailable, say because the computer hosting the copy crashes or because that part of the network is partitioned from another part), as well as the ability to access a copy that is close by, thus saving on communication costs. However, in order to realize these benefits, a protocol for keeping the copies consistent is needed, and thus induces additional costs, most notably additional communication costs.

A relatively recent approach to reducing the communication costs associated with replicated data is to use probabilistic quorums (a quorum is a subset of copies out there) [11]. This method reduces the communication cost, but causes read accesses to the shared data to return out-of-date information with a small probability. Lee and Welch [12] explored applications that could tolerate infrequent out-of-date information while exploiting the benefits of the probabilistic approach to replicated data. We propose using this approach to collect, distribute, and maintain the data sources needed for the design optimization problem.

In more detail, our proposed system architecture is the following. We have a collection of computing entities (processes), which communicate with each other by passing messages over a network. We will have software running on top of this message passing system that implements a set of shared variables using probabilistic quorums, in order to gain the benefits discussed above (increased availability and fault-tolerance).

Some of the processes monitor a data source (such as a remanufacturing facility updating processing costs) whereas others perform the computation to solve the optimization problem. Each data source writes the current value of its parameter into a unique shared variable whenever the value changes. (Reminder: at the lower layer, this "write" causes messages to be sent to the processes hosting a certain subset of the replicas designated for this virtual shared variable.) When the optimization solver decides it is time to (re)compute the solution, it reads all the data sources. (Reminder: at the lower layer, each such "read" causes messages to be exchanged between the solver process and the processes hosting a certain subset of the replicas.)

With high probability, the values read from the data sources using this software layer are the most up-to-date values. However, even if they are not up-to-date, in this application the impact on the optimal solution is expected to be minimal since the data of concern, such as remanufacturing cost estimates, are not expected to change significantly over very short time periods. This application can tolerate stale data while taking advantage of the benefits.

4 DISTRIBUTED DATA AND DESIGN DECISION MODEL

This section further builds upon the model for sustainable design introduced earlier [8]. The model employed industry average values for all parameters. Computational limits restricted the application to static inputs implementable on a single personal computer. Real world applications, however, require dynamic inputs from widely distributed data sources. Rapid expansion in computing and database

systems in recent years has resulted in a wealth of data. Keeping this in mind, this paper extends the model in various ways:

1. Linking the distributed data sources shown in Table 1 (columns 1 and 2) directly into the design decision model.
2. Updating the distributed data.
3. Reformulating the design model to accommodate widely distributed data that is being continuously updated, as shown in column 3.
4. Making the data sources fault tolerant by replicating them and using the quorums to reduce the cost of replication.

Table 1. Distributed data sources, their information and integration in design model

Distributed Database Source	**Types of Information**	**Decision Model Integration**
Material Supplier	Updated cost and availability of materials	Costs of components updated accordingly
Component manufacturer, Assembler	Updated cost of manufacturing and assembly	Costs of manufacturing and assembly updated, cost of component updated
Product take-back center operations	Updated information on time and expense of performing different operations	Updated collection, disassembly, remanufacture costs
Product take-back center: Wastewater	Updated wastewater treatment requirements	Updated disposal costs, Environmental Impacts
Customer	Preferences, willingness to pay	Objective function attribute scaling factors

Once the products are collected from the customers, four post recovery decisions at the component level can be made, defined as follows:

- **New:** The recovered component(s) cannot be reused at all and must be disposed. A second generation product would have a new component in place of the old one.
- **Reuse:** The recovered component(s) can be directly reused after minor cleaning and refurbishment.
- **Remanufacture:** The recovered component(s) requires some rework (such as milling) before it can be reused in the second generation product.
- **Recycle:** Although the recovered component(s) cannot be reused in its current form, materials (metal, plastic) can still be cost effectively extracted and reformed.

Each of the decisions described above will in turn have different operations associated with it, as listed in Table 2. These operations have different costs and environmental impacts for different components depending on the material used, weight and ease of disassembly. Moreover, performance of the next generation product will also be determined in part by these operations.

Table 2. Operations associated with four post recovery decisions

New	**Reuse**	**Remanufacture**	**Recycle**
Collection Disassembly Disposal Material processing Manufacturing Assembly	Collection	Collection Disassembly Remanufacturing Assembly	Collection Disassembly Recycling operations Manufacturing Assembly

4.1 Design Optimization Formulation

The decision model we employ is presented in equations 1-9. The major elements are described in the following paragraphs, starting with a description of the constraint equations 2-4, which embody the cause and effect relationships between design decisions and the resulting component attributes of cost, perceived age, and environmental impact. These three attributes are determined by the decision that was made regarding its use after take-back. Let d_i denote the binary design decision variable such that:

$d_1 = 1$, if the component is new, 0 otherwise.
$d_2 = 1$, if the component is reused, 0 otherwise.
$d_3 = 1$, if the component is remanufactured, 0 otherwise.
$d_4 = 1$, if the component is recycled, 0 otherwise.
Subject to: $d_1 + d_2 + d_3 + d_4 = 1$

Constraint equation 2 gives us the cost of a product for a particular customer segment p based on the component level decisions for that segment. Overall product cost is the summation of costs associated with each individual component. The cost of each component is determined by the costs incurred while performing individual operations involved in manufacturing. In the case of a new component, for example, the recovered component would have to be collected, disassembled and disposed before new material is utilized to manufacture the component. This gives us:
Cost of a component = d_1 (Cost of new component) + d_2 (Cost or reused component) + d_3(Cost of remanufactured component) + d_4(Cost of recycled component)

$$\text{Maximize } U_{total} = \sum_{p=1}^{4} f_p \frac{1}{K_p}\left[\left(K_p k_{C,p} U_{C,p}\left(C_p\right)+1\right)\left(K_p k_{A,p} U_{A,p}\left(A_p\right)+1\right)\left(K_p k_{E,p} U_{E,p}\left(E_p\right)+1\right)-1\right] \quad (1)$$

Subject to:

$$C_p = \sum_{i=1}^{s}\left[d_{1,i,p}\left(C_{8i}+\sum_{n=1}^{5} C_{ni}\right)+d_{2,i,p}\left(C_{4i}\right)+d_{3,i,p}\left(\sum_{n=3}^{6} C_{ni}\right)+d_{4,i,p}\left(C_{7i}+\sum_{n=2}^{5} C_{ni}\right)\right] \quad (2)$$

$$A_p = \overline{R}^{-1}\left(R\left(R_i,...,R_s\right)\right) \quad (3)$$

$$E_p = \sum_{i=1}^{s}\left[d_{1,i,p}\left(E_{8i}+\sum_{n=1}^{5} E_{ni}\right)+d_{2,i,p}\left(E_{4i}\right)+d_{3,i,p}\left(\sum_{n=3}^{6} E_{ni}\right)+d_{4,i,p}\left(E_{7i}+\sum_{n=2}^{5} E_{ni}\right)\right] \quad (4)$$

$$R_i = e^{-\left(\frac{d_2+0.5d_3+0.1d_4}{\theta_i}\right)^{b_i}} \quad (5)$$

$$C_{p,\min} \le C_p \le C_{p,\max} \text{ for } p = (1,\ldots,4) \quad (6)$$

$$A_{p,\min} \le A_p \le A_{p,\max} \text{ for } p = (1,\ldots,4) \quad (7)$$

$$E_{p,\min} \le E_p \le E_{p,\max} \text{ for } p = (1,\ldots,4) \quad (8)$$

$$d_{1,i,p} + d_{2,i,p} + d_{3,i,p} + d_{4,i,p} = 1 \quad (9)$$

$$d_{1,i,p}, d_{2,i,p}, d_{3,i,p}, d_{4,i,p} \in [0,1] \quad (10)$$

Where:
$C_{1,i}$ = New material acquisition costs for component i .
$C_{2,i}$ = Manufacturing/forming or remanufacturing costs for component i.
$C_{3,i}$ = Assembly costs for component , a function of the processing time required to assemble component i.
$C_{4,i}$ = Take-back costs for component i incurred by the manufacturer under product take-back legislation.
$C_{5,i}$ = Disassembly costs for component i, a function of the time to disassemble the product.
$C_{6,i}$ = Remanufacturing costs (inspection, testing, and repair or replacement) for component i.
$C_{7,i}$ = Cost of recycling operations for component i, a function of the energy required to melt the material.
$C_{8,i}$ = Disposal costs associated with placement in a landfill for component i.
$E_{1,i}$ = Environmental impact resulting from the transport of materials to the facility for materials processing.
$E_{2,i}$ = Environmental impact of manufacturing operations.
$E_{3,i}$ = Environmental impact of assembly operations.

$E_{4,i}$ = Environmental impact due to the collection and transportation for take-back purposes.
$E_{5,i}$ = Environmental impact of disassembly and separation processes.
$E_{6,i}$ = Environmental impact of remanufacturing operations including inspection, testing, and repair.
$E_{7,i}$ = Environmental impact of recycling operations including melting down used material.
$E_{8,i}$ = Environmental impact of disposal in a landfill.
p = Customer group (1 = Neutral, 2 = Technophile, 3 = Utilitarian, 4 = Green)
s = number of components in the product.
t_i = number of years component i to be installed has been in use before.
K_p = Normalizing constant for customer p.
$k_{C,p}$ = Scaling constant corresponding to cost for customer p.
$k_{A,p}$ = Scaling constant corresponding to age for customer p.
$k_{E,p}$ = Scaling constant corresponding to environmental impact for customer p.
f_p = Fraction of customers of type p.
v_i = Criticality associated with the ith component.
R = Reliability function for the computer based on the component reliabilities and failure mode assumed.
$\overline{R}$ = Reliability function for the computer based on the component reliabilities and failure mode assumed when all components are the same age.
θ_i = Characteristic life of the component i.
b_i = Slope of the Weibull reliability curve for component i.
$d_{1,i,p}$ = Fraction of components of type i the are new for customer type p.
$d_{2,i,p}$ = Fraction of components of type i the are reused for customer type p.
$d_{3,i,p}$ = Fraction of components of type i the are remanufactured for customer type p.
$d_{4,i,p}$ = Fraction of components of type i the are recycled for customer type p.

Equation 3 shows how the individual ages of components are aggregated into the perceived age of the whole product. The model allows for components that are recovered from/utilized in the same product to have different ages. Age (rather than reliability) is used as a measure of performance because it is easier for customers to relate performance to age rather than physical reliability. This is important in the case of electronic components, since they become obsolete quickly. It also simplifies finding the overall product attributes from those of individual components as discussed below. Age of a collected component is given by how long the customer owned it before it was returned. If a component is reused or remanufactured after collection, its reported age would be how long it has been in the market before. On the other hand if a new or recycled (manufactured from recycled materials) component is supplied it would have an age of 0 years. In practice, however, products made from recycled components perform slightly worse than new, and remanufactured components perform slightly worse still. To account for these we assume that recycling recovers 90% (and not 100%) of the value while remanufacturing recovers 50% of the value of the component. This gives us:
Age of a component = (d_2 + 0.5d_3 + 0.1d_4) (Time the component has been in use)
To find the overall perceived age of the product, first the product reliability is found using the failure mode information. The failure mode we assume is when any of the critical components (motherboard, hard drive or video card) fail, or any three of the other components fail simultaneously. A reliability function, R (equation 3), is created using this failure mode information to map the age of the components to product reliability. Once the ages of the components are known, the product reliability is found and mapped back to the same function when all the components are assumed to be the same age. Finding this inverse gives the age of the product aligned with that of a product that has never been disassembled (all components are the same age).
Equation 4 shows how the total environmental impact is calculated from the environmental impacts of each operation performed on each individual component. For example the bulk of the environmental impact when a component is reused is its collection and transportation. On the other hand when a new component is to be manufactured, the recovered component would have to be collected, disassembled and disposed before new material is utilized to manufacture the new component. The environmental impacts of the operations are summed to arrive at the environmental impact of the component:
EI of a component = d_1 (EI of new component) + d_2 (EI or reused component) + d_3(EI of remanufactured component) + d_4(EI of recycled component)

4.2 Objective Function

The objective function shown in equation 1 maximizes the sum of overall utilities of all the customer groups. Market segmentation is defined on the basis of the customers' preference for a certain attribute

over others. The preference for an attribute is reflected in the scaling constant one typically assesses during the standard lottery method. The values can also be determined from surveys of customers returning the products. In the case study that follows, this is implemented using a randomly chosen sample of customers. The different groups are defined as follows:

Table 3. Market segmentation based on relative preference for attributes

Customer group	Preference relation
Technophile	$k_{age} \geq 1.25\max(k_{cost}, k_{env})$
Utilitarian	$k_{cost} \geq 1.25\max(k_{age}, k_{env})$
Green	$k_{env} \geq 1.25\max(k_{cost}, k_{age})$
Neutral	Everyone else

The classifications shown above are mutually exclusive and collectively exhaustive. In this study we also account for the relative number of customers of a certain type in the population. This is calculated by finding the fraction of customers of each type in the chosen sample size. The fractions are extrapolated using the overall demand to find how many products of a certain type are needed.
Different utility functions for the customer groups are also defined. The utility functions for the range of values of an attribute that a customer is willing to buy the product are assumed linear. The acceptable ranges for individual customer segments are also acquired from surveys of customer returning the products. When combining the ranges acquired from different customers, 90% confidence intervals are used assuming upper and lower bounds are normally distributed. This was done to avoid being overly influenced by customers that provide extreme bounds.
Once the acceptable attribute ranges and scaling constants are known, utilities for each customer group can be calculated. The overall utility for each customer group considers the combined cost, age and environmental utilities of that group. In order to reflect the dynamic nature of willingness-to-pay as one moves through the feasible region, multiattribute utility analysis is used to compute the overall utility of the product portfolio. The multiplicative form of the multiattribute utility function is used. The utilities for different groups are aggregated using a linear additive function using the fractions of customers of each type. In equation 1, f_p's, where p = 1,2,3,4 are the fractions of neutrals, technophiles, utilitarians and greens in the customer base. The decision problem is to maximize this weighted utility over the customer groups with component level decisions as decision variables. The authors recognize the difficult issues involved in the aggregation of utility. In this case, we assume the group members and range of outcomes are such that the difficulties are minimized.

5 CASE STUDY

As mentioned earlier, replication allows for highly available data which is essential in cases where some of the servers are busy or not available. In this section we provide a case study to show how shared memory can be used in a product take-back environment.

5.1 System architecture and formulation

The system architecture is shown in Figure 2. Five distributed data sources provide information about the product attributes (manufacturing facilities, indicated in green) and customer preferences (indicated in red). The designer (manufacturer) reads from the shared memory the data provided by the distributed sources. The computations that follow aim at solving the optimization problem that maximizes total portfolio utility of the customers. The decisions are whether to reuse, remanufacture or recycle a recovered component or simply discard it and replace it with a new one. The customer can also read and perform an optimization to find the best reuse decisions.
Each data source writes the current value of its parameter into a unique shared variable whenever the value changes. This *write* causes messages to be sent to the processes hosting a certain subset of the

replicas designated for this virtual shared variable. When the optimization solver decides it is time to compute the solution, it reads all the data sources. This *read* causes messages to be exchanged between the solver process and the processes hosting a certain subset of the replicas.

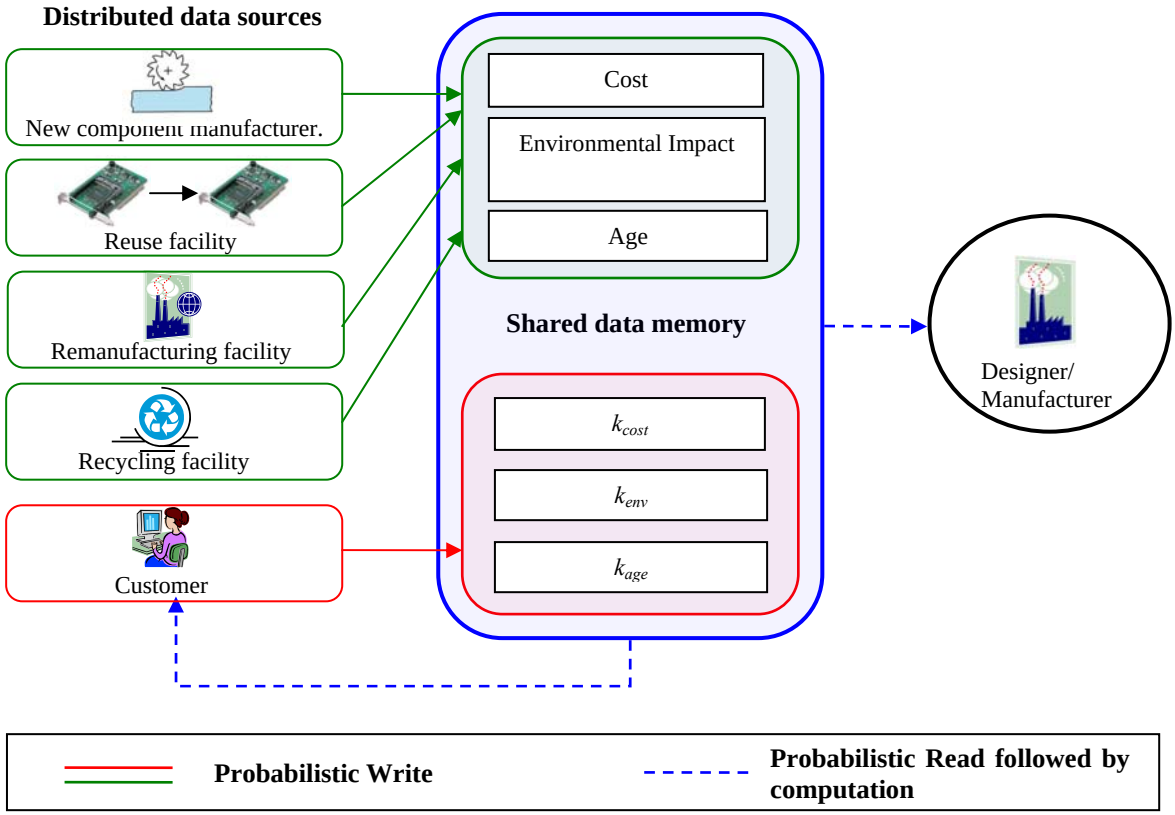

Figure 2: Schematic of the shared memory implementation for distributed data sources

We want to ensure that that the values read from the data sources are the most up-to-date values with high probability. However, even if they are not up-to-date they need to have a very high probability of being from the immediately previous time stamp. Our argument is that it would not affect the results significantly since the attributes (cost and environmental impact) do not drastically change over a short period of time. If we store each shared variable at 36 replicas and have a quorum size of 10 (so that each read or write only accesses 10 randomly chosen of the 36 replicas), the probability of reading an outdated value is 0.02. In order to guarantee that we never get stale data, with 36 total replicas, we would have to access at least 19 replicas in each read or write, so we are realizing a savings of about 50% in the communication cost by using a quorum size of 10. Furthermore, comparing with data in Lee and Welch, [12], we estimate that the probability of getting a value that is outdated by more than one timestamp is essentially 0.

In our case study we simulate the evolution of data over ten time periods. The costs include those incurred in acquiring raw materials and the manufacturing and/or remanufacturing operations shown in Figure 1. To investigate the effect of stale data acquired from the shared memory, costs of electricity, and raw materials like metals and plastics were updated using historical trends. Costs of performing individual operations like remanufacturing, recycling etc. on components were decreased steadily by 5% in every cycle. This was done to depict improving manufacturing methods and economies of scale as remanufacturing volume increases. In all, 25 entries are subject to updating each cycle, 15 of which correspond to the raw materials and 10 correspond to costs associated with processing and manufacturing/remanufacturing operations. Environmental impacts resulting from each operation were calculated using commercially available software Simapro.

The acceptable ranges of attributes that were assumed for the three customer groups are shown in Table 4. We can see that technophiles perceive cheap products as having low reliability (greater age)

therefore have a high minimum cost of $400 which they are willing to pay. Similarly, the utilitarians are only interested in low cost alternatives while environmentally conscious green customers have a wide range of cost ($100 - $1000) they are willing to pay. Technophiles also require high performance (lesser age) in their products and therefore would not accept products that are perceived to be older than 1 year. Utilitarians on the other hand require most value for their money. Green customers have a low upper limit of environmental impacts compared to the other customer groups at 600 millipoints. Neutrals are assumed to have preferences that lie between those of other customers and do not focus on any one specific attribute.

Table 4. Feasible attribute ranges for each customer group

Customer Group	**Feasible attribute ranges**		
	Cost $C_{p,\min} \le C_p \le C_{p,\max}$	**Age** $A_{p,\min} \le A_p \le A_{p,\max}$	**Environmental Impact** $E_{p,\min} \le E_p \le E_{p,\max}$
Technophiles	$400 - $1000	0 – 1 yrs	333 – 1275 mpt
Utilitarians	$100 - $600	0 – 2 yrs	333 – 800 mpt
Greens	$100 - $1000	0 – 3 yrs	333 – 600 mpt
Neutrals	$100 - $1000	0 – 3 yrs	333 – 800 mpt

Each customer segment's willingness to make tradeoffs is captured by the scaling constants shown in Table 5. Technophiles are willing to tradeoff low cost and low environmental impact for better performance (low age). Utilitarians and Greens on the other have high scaling constants associated with cost and environmental impact respectively. The neutral customer group assigns an equal scaling constant value of 0.5 to each attribute. The individual attribute utility functions are assumed to be linear and the ranges within which they are defined are kept constant. The fractions of technophiles, utilitarians, greens and neutrals (f_p's) in the entire customer base are assumed to be 0.1, 0.4, 0.1 and 0.4 respectively. These fractions are used to combine the multiattribute utilities of individual customer groups into a total product portfolio utility. In summary, the optimization problem is to find the decision variables corresponding to reuse decisions (new, reused, remanufactured or recycled) that maximize the multiattribute utility of the product portfolio as given by equation 1.

Table 5. Independent scaling constants and normalizing parameter

Customer Group	**Independent scaling factors**			**Normalizing Parameter, K_p**
	Cost	**Age**	**Environmental Impact**	
Technophiles	0.30	0.80	0.10	-0.596
Utilitarians	0.70	0.45	0.35	-0.794
Greens	0.15	0.15	0.85	-0.562
Neutrals	0.50	0.50	0.50	-0.764

5.2 Results

Figure 3 shows the optimal portfolio utility over 10 reads of the data for two cases. The "perfect information" plot (square shape) shows utility as the cost of raw materials and operations involved in reusing, remanufacturing and recycling change. This plot assumes the designer has continuous access to the most up to date, perfect information. The "with probabilistic quorums" plot (diamond shape) results from a simulation of the shared memory implementation for distributed data sources described earlier, allowing the possibility of stale data. Any of the values in each cycle could be outdated, providing stale information. As calculated before for a quorum size of 10 with 36 replicas of variables, the probability of getting an outdated data value (from an immediately preceding timestamp) is 0.02. The results illustrate that the calculated utilities are very close to those based on perfect information.

Figures 4 and 5 show the optimal cost and performance (age) for the neutral customer group. As before, the square shapes depict results based on perfect information, and diamond shapes depict results using probabilistic quorums which allow a small probability of stale information. It can be clearly seen that the perfect information and probabilistic quorum values are very close for both cost

and age. The decrease in cost of the computer is attributable to the manufacturer realizing economies of scale as time progresses.

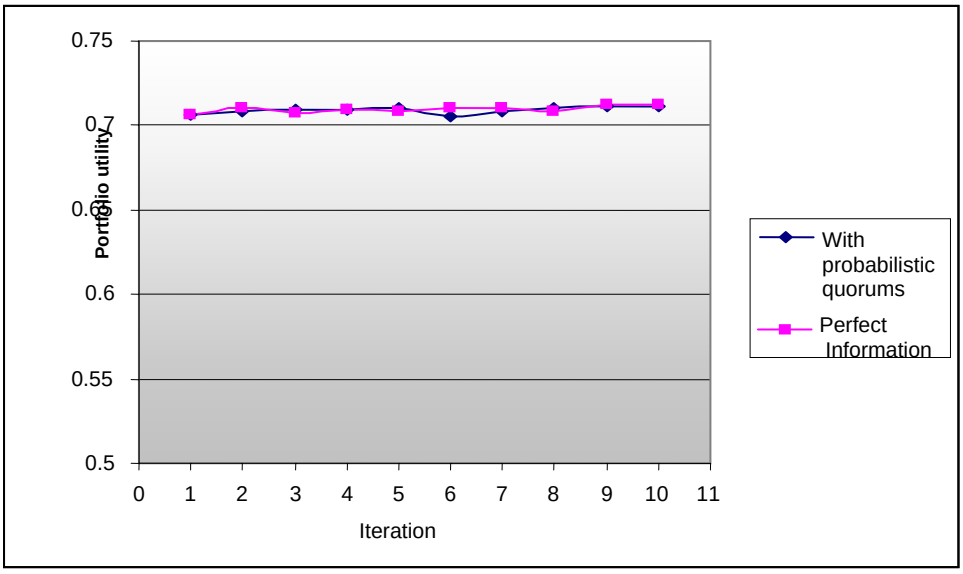

Figure 3: Comparing the maximized utility computed with 1) probabilistic quorums (information that has a small probability of being stale, and 2) perfect information.

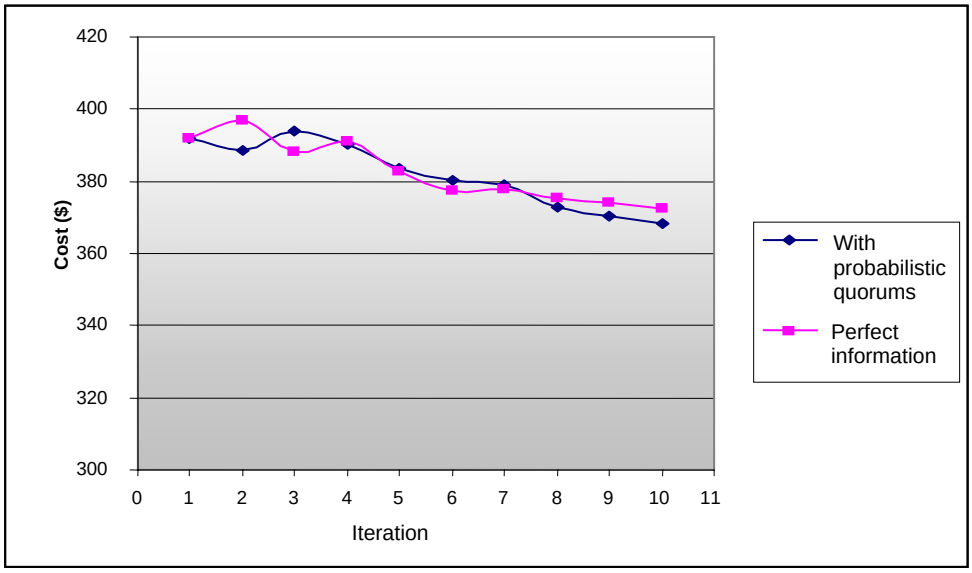

Figure 4: Cost of the computer for the optimal configuration for the neutral customer computed under 1) probabilistic quorums and 2) perfect information.

We just demonstrated that allowing a small probability of stale information does not affect the results substantially. One may be tempted to think that further reduction in communication costs could be realized if we make the probability of getting stale information even higher (by making the quorum size smaller). To demonstrate that this could lead to suboptimal decisions we could compare the results from two cases: 1) Quorum size is large enough so that perfect information is available every iteration and 2) Quorum size is so small that the information is practically not updated at all. One way of doing this could be to compare the decisions from the first and final iterations, as shown in Table 6, which shows that the optimal decisions have changed substantially for the four customer groups.

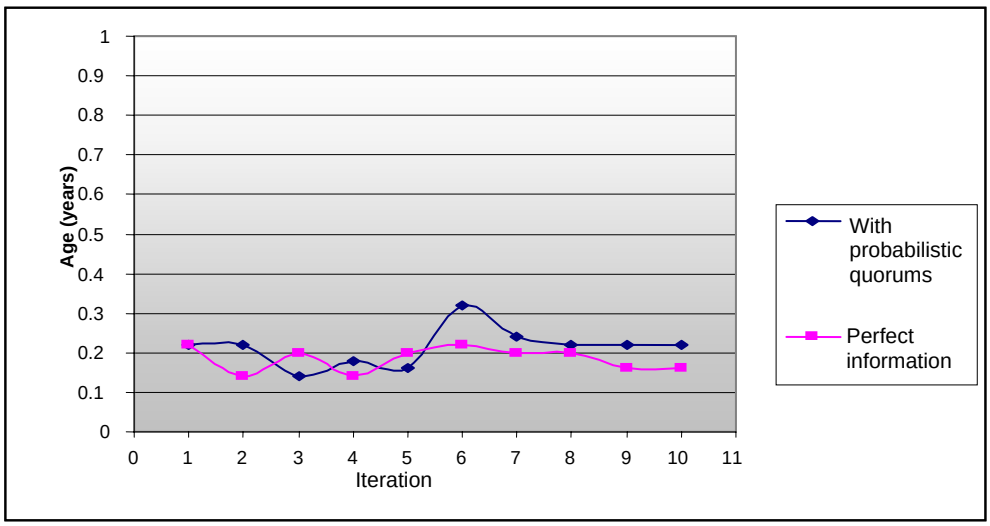

Figure 5: *Age of the computer for the optimal configuration for the neutral customer computed under 1) probabilistic quorums and 2) perfect information.*

Table 6: Maximized portfolio utility and corresponding decisions in the first and final iterations

	First Iteration					Final Iteration			
	Tech	**Utilitarian**	**Green**	**Neutral**		**Tech**	**Utilitarian**	**Green**	**Neutral**
Monitor	Reuse	Reuse	Reuse	Reuse		Reuse	Reuse	Reuse	Reuse
Floppy Drive	New	Recycle	Reuse	New		Recycle	Recycle	Reuse	New
Keyboard	New	Recycle	Reman.	Recycle		New	Recycle	Reman.	Recycle
Hard Drive	New	Recycle	Reuse	New		New	Recycle	Recycle	New
CD-ROM	Recycle	Recycle	Reman.	Recycle		Recycle	Recycle	Reman.	New
Motherboard	New	Reuse	Reuse	Recycle		New	Reuse	Reuse	Recycle
Power Supply	Recycle	Reuse	Reuse	Reuse		Recycle	Reuse	Reuse	Reuse
Sound Card	Reman.	Recycle	Reman.	New		New	Recycle	Reman.	New
Video Card	Reuse	Reman.	Reuse	New		New	New	New	New
Modem	Recycle	Reuse	Reman.	Reuse		Recycle	Recycle	Reman.	New
Cables	Recycle	Reman.	Reman.	Recycle		Recycle	Recycle	Reuse	Recycle
Housing	Recycle	Recycle	Recycle	Recycle		Recycle	Reuse	Reuse	Recycle
Cost ($)	547.72	184.8	170.11	391.84		565.99	183.21	173.96	372.28
Age (years)	0.02	1.6	2.98	0.22		0.03	1.59	2.61	0.16
Env. Impact (mpt)	926	610.5	537.7	657.4		921.3	607.7	544	659.5
Utility	0.92	0.70	0.32	0.75		0.91	0.71	0.32	0.77
	Portfolio utility					**Portfolio utility**			
	0.706					0.711			
Effect of applying the decisions from the first iteration when data has changed.						0.92	0.70	0.33	0.76
						Portfolio utility			
						0.710			

Elements highlighted in yellow indicate differences in the optimal decision variables. In addition, the

product portfolio utility has increased from 0.706 to 0.711. If the decisions were prescribed using information from only the first cycle (no updating) there would be a decrease in utility as shown. Even though there is improvement compared to the first cycle because the cost decreases, it is not optimal for the most current data, as shown.

6 SUMMARY

Our results show that updating information frequently helps designers make better decisions. We also demonstrated that a small probability of obtaining outdated information did not affect the results substantially. It was seen that a shared memory approach can be easily used for data that does not change drastically over a short period of time. The deviations from the assumption of perfect information were shown to be minimal. Information acquired using multiple replicas of data and probabilistic read and write results in reduced communication costs. By using a quorum size of 10 instead of 19 (which would guarantee up-to-date information) we reduced communication costs by about 50%. Having multiple copies of data also allows for robustness against partial system failure. It was also demonstrated that reusing components reduces cost and environmental impacts over several product lifecycles. In the case where only small components are replaced with new ones, performance can be improved with little effect on cost and environmental impact. As manufacturers devise better ways to reuse and remanufacture components, they can compete effectively by introducing selected new components, as demonstrated in our case study.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grants DMI-0500265 and DMI-05-00464. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] "Directive 2002/96/EC of the European Parliament and of the Council of 27 January 2003 on Waste Electrical and Electronic Equipment". Official Journal of the European Union, L37/28 and L37/29.

[2] "Directive 2000/53/EC of the European Parliament and of the Council of 18 September 2000 on end-of life vehicles" Official Journal of the European Communities. Article 7.

[3] Ferrer, G., 1997, "The Economics of Personal Computer Remanufacturing," *Resources, Conservation and Recycling* 21, pp 79-108.

[4] Sandborn, A., and Murphy, C. F., 1999, "A Model for Optimizing the Assembly and Disassembly of Electronic Systems," *IEEE Trans. on Electronics Packaging Manufacturing,* vol. 22, no. 2, pp. 105-117.

[5] Campbell, M.I. and A. Hasan. Design Evaluation Method for the Disassembly of Electronic Equipment, *ICED '03*, Stockholm, August 2003.

[6] Bhamra T.A., S. Evans, T.C. McAloone, M. Simon, S. Poole, A. Sweatman, 1999, "Integrating Environmental Decisions into the Product Development Process: Part 1," *Ecodesign* , p. 329.

[7] Thurston, D. L., "Design Evaluation and Optimization of Multiple Attributes*," Proc. of 1991 International Conference on Engineering Design - ICED 91* (Zurich, Switzerland, August 1991).

[8] Mangun, D. and D. L. Thurston, 2002, "Incorporating Component Reuse, Remanufacture, and Recycle into Product Portfolio Design," *IEEE Transactions on Engineering Management*, Vol. 49, No. 4.

[9] Thurston, D. and J. de la Torre, "Leasing and Extended Producer Responsibility for Personal Computer Component Reuse", *International Journal of Environment and Pollution,* Vol. 29, No. 1/2/3, pp. 104-126.

[10] Gifford D.K., "Weighted Voting for Replicated Data," Proceedings of the 7th ACM Symposium on Operating System Principles, pp. 150 - 162, 1979.

[11] Malkhi D., Reiter M., Wool A. and Wright R., "Probabilistic Quorum Systems", Information and Computation, vol. 170, no. 2, pp. 184-206, November 2001.

[12] Lee H. and Welch J.L., "Randomized Registers and Iterative Algorithms," Distributed Computing, vol. 17, no. 3, pp. 209-221, March 2005.

Contact: D. L. Thurston
University of Illinois at Urbana-Champaign
Department of Industrial and Enterprise Systems Engineering
104 S. Mathews

Urbana, IL
USA
Phone 217-333-6456
Fax 217-244-5705
thurston@uiuc.edu
www.iese.uiuc.edu