Implications of Buyer Decision Theory for Design of eCommerce Websites

By

Barry G. Silverman[1], Mintu Bachann[2], Khaled Al-Akharas[2]
1- Dept. of Systems Engineering, University of Pennsylvania, 2- Equalfooting.com
contact: barryg@seas.upenn.edu

June 2001

" . . .it came up with streams and streams of information and it just took ages to scroll through it all -- and it never came up with anything particularly useful. . .I then tried to enter words that were more specific to what we wanted, but in the end I just gave up because I couldn't find anything."
-Anonymous user in [2]

"Last month we finally had an intern look at the most common search queries …the right answer for the top query wasn't showing up until item 47."
-Anonymous Shopping Site Executive in [13]

ABSTRACT

In the rush to open their website, e-commerce sites too often fail to support buyer decision making and search, resulting in a loss of sale and the customer's repeat business. This paper reviews why this occurs and the failure of many B2C and B2B website executives to understand that appropriate decision support and search technology can't be fully bought off-the-shelf. Our contention is that significant investment and effort is required at any given website in order to create the decision support and search agents needed to properly support buyer decision making. We provide a framework to guide such effort (derived from buyer behavior choice theory); review the open problems that e-catalog sites pose to the framework and to existing search engine technology; discuss underlying design principles and guidelines; validate the framework and guidelines with a case study; and discuss lessons learned and steps needed to better support buyer decision behavior in the future. Future needs are also pinpointed.

1) INTRODUCTION

The rapid rise of e-commerce is impressive – over half of today's 80 million web users shop for or buy products online, and business to business purchasing is expected to rapidly eclipse that level [1]. However, in the rush to provide online presence, many eMarket sites have been built quickly, with little infrastructure and capabilities needed to run such an e-business. As we all know only too well, browsing, searching, and buying via online web catalogs can be a time consuming, frustrating task. BCG, for example, reports that over 80% of web shoppers have at some point left eMarkets without finding what they want and that 23% of all attempted e-shopping transactions end in failure [1]. Four of the top five failure modes are search-related (i.e., page loading times, couldn't find product, system crashed, had to call customer service) although some of the blame

needs to be shared by other causes as well, a few of which are internet delays, overall website design, and reliability.

This paper explores only those portions of these failure modes that are due to the functioning of the Decision Support System (DSS) and search software. In particular we examine DSS that operates at shopping sites. While the results in this paper apply to all shopping sites, they are particularly pertinent to sites that are mergers of multiple catalogs, sometimes called gateways, e-exchanges, or eMarkets. We refer to the dot com that creates the exchange as the market maker. The reader will see that the problems of merging multiple vendor catalogs into a seamlessly integrated and easily searchable, virtual shopping mall turn out to be formidable. In looking for a solution, this paper will postulate a design framework for DSS in online shopping sites and attempt a proof of concept test of the framework and infer lessons learned from a case study of an existing eMarket site. A single case study is insufficient to fully validate the framework, hence we only offer a single data point concept assessment.

1.1) Who are the Sellers

Market places on the web may be single vendors but increasingly they involve aggregator sites [19], online virtual interfaces that merge potentially 1,000s of vendors into a seemingly single marketplace. Exchanges vest control in the buyer since they foster cross-vendor shopping. To be successful, exchange sites must pay extra attention to the design of the DSS they provide for the buyers to browse and search their site. Benefits to the vendors arise when the exchange draws in and retains numerous prospects and potential customers that might never have visited each of the individual vendors' sites. DSS can vital play a role in this process.

Such sites exist today as business to consumer (B2C) mega-stores and as business to business (B2B) hubs which aggregate numerous small suppliers horizontally. B2C examples are Amazon.com (books, electronics, tools, garden supplies, etc.), Priceline.com (food, airline tickets, shows), and others. B2B examples are EqualFooting.com (operations & maintenance supplies), NECX electronics exchange (semi-conductors, electronic components), etc. In the B2B cases, the sellers often are also buyers of other sellers' goods. Typically the sellers participate in these markets to eliminate the middleman, to gain new customers, to keep up with competitors, and/or to appease shareholders. Many sellers are themselves new to information technology and often rely on the market-maker to try and automate and integrate their catalog as best as possible from the raw data they provide.

Forrester Research interviewed executives at 50 e-commerce sites in 2000 and found that while 90% state search is extremely or very important, most have invested minimal or no money or time in developing a good search engine, and over half admit they don't even know how their engine is used or whether customers are unhappy with it [13]. The sites failed a bank of basic search tests that Forrester administered [12,13].

1.2) Who Are the Buyers

Online shoppers to date include almost 15% of the country. The mass market has yet to show up in either the B2C or B2B forums, but projections indicate this will occur

rapidly over the next few years. To date we do know that the process of finding information via search engines is simply too complicated for internet-naive people to use without very high levels of support. And, some users who have had internet access for over a year (and who claim to 'surf' on a regular basis) are still having problems using search engines [2]. For example, Neilsen [3] gave users the task:

> *You have the following pets:*
> *- cats*
> *- dogs*
> *Find information about your pets.*

Almost all users enter the query `cats AND dogs`. They typically find nothing, since the site does not include pages that mention both animals. Upon encountering a "no hits found" message, the vast majority of users concluded that there was no information available about these pets and departed from the site. Even sophisticated web-searchers made this mistake initially, though they soon recovered. In short, one must be very cautious in designing the search experience to support such users, be they in B2C or B2B marketplaces.

1.3) What is an Online Catalog

Before going further, it is worth elaborating on the definition of an online catalog and some of the design challenges it represents. An e-commerce catalog is the heart of a shopping site and it holds information on all the products one can buy at that site. The catalog is browsable like other website contents, but unlike the other contents, it usually is stored in a relational database product as are the transactions such as bid, buy, ship, etc. On the face of it, the reason a relational database is used is that the catalog is far more structured than the HTML documents typically found on a website. However, the degree of structure is relative, and most product catalogs do not have nearly enough structure for DSS and search to work at their optimum [16].

The basic logical structure of a product or item catalog may be described as four sets of Relations, $R = \{R1, R2, R3, R4\}$:

Product Hierarchies (*R1*) – These are the fields supplied by the website when users try to browse the catalog. As a result, $R1 = <\mathbf{C_1}, \ldots, \mathbf{C_N}>$ is a tuple of multi-level trees that taxonomize the contents of the catalog and whose leaf nodes point to actual products in the catalog. When merging each of the $\mathbf{C_N}$ catalogs of multiple vendors on a dynamic, continuing basis, market makers encounter almost overwhelming taxonomic challenges to constructing *R1*, which in turn pose browsing design concerns – obstacles we return to in the case study.

Product Descriptions (*R2*) – This relation is a tuple, $R2 = <\mathbf{D_1}, \ldots, \mathbf{D_K}>$, of k=1,K free text descriptions that suppliers create and which hold the information about products that suppliers believe users need. The D_K are the document fragments utilized by search engines to support user queries and product search commands. Interestingly, while catalogs include a field called “product name”, most suppliers omit this field believing the “description” field is sufficient. When merging 100,000s and 1,000,000s of products, market-makers have no resources to supply this or other missing information, or to standardize free text descriptions the suppliers provide in the D_K. As we will see

later, this makes for some formidable search engine design challenges – ones the field has yet to entirely overcome.

Product Attributes and Values (*R3*) – The catalogs also hold numerous parameters about each product (e.g., color, weight, length, manufacturer, price, condition (new, used), availability, reviews, etc.). This is the information that is used to support parameter search and sorts such as by price or size or location. A universal feature of catalogs is that there are many products, the product lines are continually changing, and each of the 1,000s of leaf node categories of products has a different set of attributes. As a result, attributes are not stored as fields of a table. Rather, the m=1,M categories of attribute names are stored as data items as are their value settings. Thus $R3 = <<\mathbf{A_{31}, V_{31}}>, \ldots, < \mathbf{A_{3M}, V_{3M}}>>$ where A and V are vectors of a-v pairs that are equi-length for any given category**.** This and large numbers of unfilled in attribute fields are stumbling blocks few DSS designer have yet to fully eliminate in trying to support attribute or parameter search.

Transactional Fields (*R4*) – A final point about catalogs is that they must also be designed to support the full range of transactional activities such as customer registration and profiling, product purchases, purchase histories, shipping requests and status, payment choices and status, bids placed and received, and the like. That is, the database serves as the underlying organizing vehicle for the processes, tasks, and workflow surrounding each product. Successful catalogs hide these fields behind highly usable, visual metaphors such as the shopping cart, the activity center (where you can track your purchases and bids), and other web pages. These fields and their visual interfaces must be designed with DSS and user decision processes in mind. Too often, one encounters shopping sites that needlessly interrupt users' buying steps, that fail to support how customers want to proceed, and/or that are unforgiving of user error. Overcoming these problems requires research and investment in better DSS technology.

1.4 What is a Decision Support System

A decision support system (DSS) is an interactive information system that provides information, models, and data manipulation tools to help make decisions in semistructured and unstructured situations where no one knows exactly how the decision should be made. The traditional DSS approach includes interactive problem solving, direct use of models, and user-controllable methods for displaying and analyzing data and formulating and evaluating alternative decisions [20,33]. This approach grew out of dissatisfaction with the traditional limitations of Transaction Processing Systems and Management information Systems. The former focused on record keeping and control of repetitive clerical processes. The later provided reports for decision makers, but were often inflexible and unable to produce the information in a form in which people could use it effectively. In contrast, DSS were intended to support the mental processes of people doing largely analytical work in less structured situations with unclear criteria for success. DSSs are typically designed to solve the structured parts of the problem and help isolate places where judgment and experience are required. The broad spectrum of information systems with DSS label range from general tools such as spreadsheets, data analysis, and graphics packages to highly customized simulation models or knowledge based advisor system focusing on a specific situation.

In online shopping sites, buyers have decisions to work out and tasks to perform that require DSS software. In the next section we review the mental processes of buyers in detail. Here however, we are more generally focused on how DSS can be brought to bear in buying decision making. To address this we introduce Table 1, which illustrates the levels of DSS one typically adds to a (B2C or B2B) shopping website. These range from minimal DSS (level 1) to sophisticated (level 3). The levels also parallel the evolution of shopping DSS since the mid 90's though one still finds late adoptors or low budget sites coming in at level 1, and no sites have yet fully developed level 3.

Level 1 DSS tend to use off-the-shelf general-purpose software to set up the shopping pages and the simplest of keyword-based search and browse technologies. These sites are "user pull" based and cost very little to construct. Generally, they are executed to address web-presence issues, and they offer little in the way of helping the buyer with anything but information access and the barest minimum of purchasing functionality.

At the next level are a series of efforts to better understand buyer mental processes and steps during a transaction and to offer numerous default settings and templates to help with the more well structured steps. DSS surveyed in the late 1990s by Miles et al, for example, focus on how to support the steps involved in comparing brands and products. Only a short while later another surveyed by Guttman et al, showed systems trying to extend this to the buying/negotiation/shipping steps. In the current paper, we integrate these approaches and extend the DSS further to cover still more decisions process such as finances, online help, and error management as well as natural language-based search.

At the highest level are the most recent shopping site features that have been developed in the current millennium. Here we see a shift to support the long-term relationship that the prospect and buyer have with the site via knowledge based prompting both within a transaction (as with level 2 approach) and across transactions. The current paper incorporates these features into its framework and suggests ways to push the state of the practice still further. To better understand this we turn now to a more in-depth look at buyer behavior theory and how DSS features seek to support the individual molar processes of buyer behavior and relationships.

Table 1 Alternate Forms That DSS May Take For Supporting Buyer Decision Making At Shopping Sites And Market Exchanges

Level Of DSS	DSS Characteristics	Examples /Frameworks
1.Access Focused	Scope: • All-Purpose Web Server & Search Tools Useful For Any Domain (Not Just Shopping) Prime Features: • Linear Search & Browse • Key Word Search • Web Server Cost & Effort: Turn Key Solutions Offered By Many Vendors But Many Shopping Website Functions And Pages Must Be Programmed	Very Early Internet Era Shopping Sites Circa 1995
2.Trans-action Focused	Scope: • Shopping Focused Tool Set • Mental Model Of Buyer • Guided Choices Prime Features: • Shopping Site Data Structures And Web Server Applications • Templates & Default Settings For Product & Brand Selection Process • Templates & Default Settings Extended To Shipping/Buying/Bidding • Template & Default Settings Extended To Financing And Help Desk Cost & Effort: Focuses On Catalog Content Integration (And Quality) And Numerous GUI Issues	Miles et al, Survey Of 13 Web Sites [25] Guttman et al, Survey Of Cross Market Site Agents [16] Silverman et al (This Paper)
3.Rel-lationship Focused	Scope: • Customized Tuning To Prospect & Customer Mental Model At This Site • Knowledge Based Prompting Prime Features: • Trouble Management Systems (Human And Computerized Customer Relationship Management) • Cross Session Consumer Preferences And Personalization (Life Cycle Management) • Reminding/ Advertising/Extended Supply Chain (Client Organization Support Of Purchasing Function) • Natural Language "DO What I Mean" Cost & Effort: Must Embellish Vendor Offerings, Add Warehouse, And Integrate Several Vendors' Applications	Many B2C Sites Attempt To Offer These At Present But Few B2B Sites Yet Do Silverman et al, (This Paper)

2) FRAMEWORK FOR DESIGN OF A DECISION SUPPORT SYSTEM FOR ONLINE SHOPPING

This section examines the buyer's transaction and relationship based DSS literature and models more closely. Unfortunately, there is no single source that one can turn to which will guide the design of the decision support functions that need to be built for effective support of users performing e-commerce tasks. Instead there are numerous sources, each presenting part of the solution: e.g., see [4-10] among many others. What is needed is a synthesis of guidance, a DSS developer's design guidance framework. In this section we present such a synthesis and explain its derivation. In subsequent sections

we attempt to evaluate a portion of the framework via a case implementation. It will be seen that the framework poses a larger research problem than the one paper can do justice to.

Let us begin by examining the work of Miles et al. [4]. They survey the buyer behavior literature and isolate an insightful three stage model of buyer behavior including: (1) initial identification and subsequent management of search criteria that happens as the search proceeds, (2) search (via browsing, engine, or other method) for a product based on the current criteria set – this is an information collection and intelligence building stage in which more is learned about the criteria, the products' attributes, and the merchants of those products, and (3) comparison of products leading to a choice, or to a decision to abandon the search. Miles et al. point out that these are three categories of goal driven behavior that are not readily modeled as a hierarchy or sequence. The precedence of the stages is determined by on-screen information, multiple product searches might be interleaved (e.g., looking for a video camera and a television), and management of criteria and comparative results for one search might affect those of another. In short, there is no simple decision algorithm that one can deduce for this process.

Instead, Miles et al. review 13 alternative types of shopping websites and use the lessons learned to construct a framework of design alternatives for DSS in online shopping. We summarize their framework in the innermost dashed box of Figure 1. Here one can see the 3 stages of buyer behavior across the top. The lower boxes express the range of DSS design options available to site developers. Thus alternative styles of product representations and form of parameter information presentations can be chosen to support the criteria management stage. Likewise the search stage is variously supported by sites that are browse- vs. search- vs. assistant-based (metaphor), and by different search technologies (keyword, concept, parameter, or natural language). Finally, in the compare stage developers choose DSS alternatives in terms of the scope (compare across multi-vendors at once?) and mode of information the user can view. This inner dashed box of Figure 1 takes a few liberties with the Miles et al. framework. For example, they omitted "Trouble Management," which is a vital DSS design feature given the frustrations buyers are experiencing today on websites. In addition to adding this box, we have added a few bullets here and there to other boxes that modernize Miles et al. such as natural language. As another example, exchanges did not exist at the time of their research, and we now have the potential of designing not only for exchanges, but also across exchanges (see "X-exchange" bullet). Aside from these updates, the basic framework is a useful one.

Another subtle difference of opinion is that Miles et al. indicated the bullets in each box of the framework tend to be mutually exclusive. For example, they try to label a site as search vs. browse metaphor, or keyword vs. parameter search. The benefit of time has proven this to be a weak assumption, and many sites today increasingly seek to support multiple features of the design space simultaneously. This is because no one approach works all the time, and users have different cognitive styles and support needs.

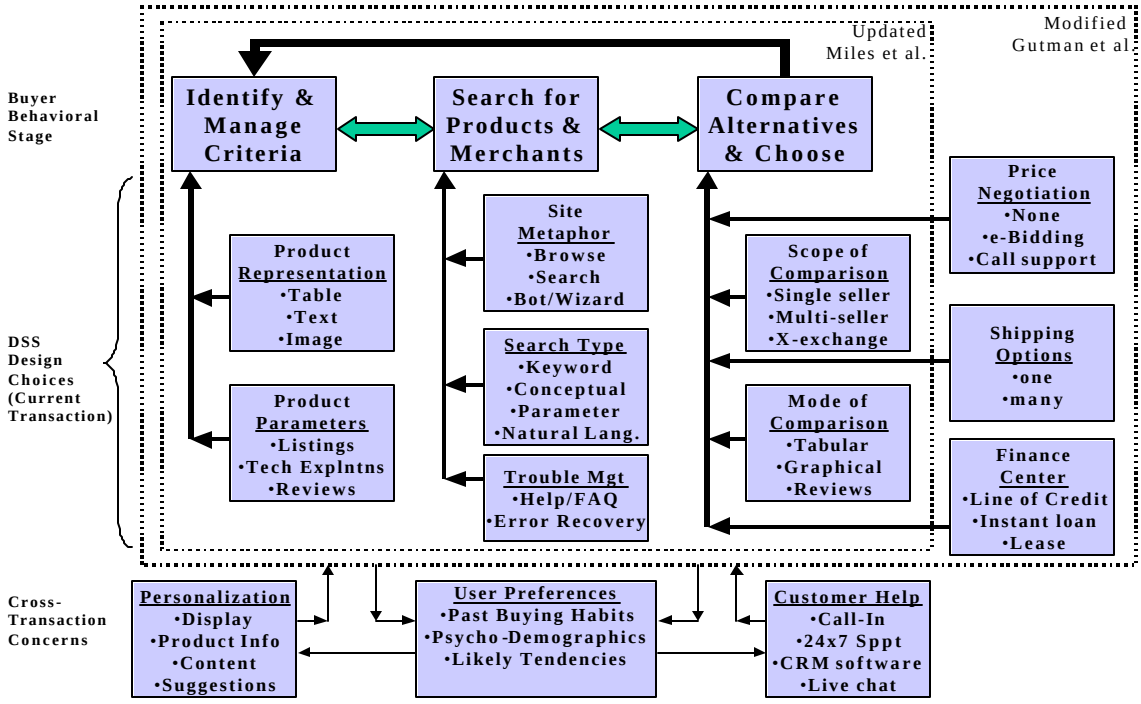


Figure 1 – Overview of Design Space for Online Buyer Decision Support

The Miles et al. model of behavioral stages also ignores some of the later stages of decision making such as price negotiation, how to get the product (shipping), and how to pay for it (finance). Guttman et al. [5] survey the buyer behavior literature and summarize it in terms of a 6 stage model that encompasses the Miles initial stages, these three later stages just mentioned, and a final stage for user evaluation of the service they got during the purchase. We plot most of these extra stages (all except post-purchase evaluation) along the right side of Figure 1 – outer dashed box. Here we show them as DSS design choices since how they are implemented effects the purchase choice, and there are alternative ways to design them (i.e., choices of services to offer at the site). Miles et al. does include the bidding and negotiation options in their design space framework, but they omit the other boxes. We think that is a serious omission and offer this framework as a revision. Guttman et al. [5] identified these as stages, but we rightly feel they are subordinate to the comparison stage since they impact the buyer's decision of which product to select and which merchant to buy from. In that sense they are clearly alternative services that the site developer might want to factor into their DSS design. Finally, Guttman et al.'s finance stage has been reinterpreted here in the following sense. That is, the idea of instant credit and loans over the web did not exist at the time of their survey and so they meant similarly "paying options." We extend this here to include the newer financial services that market exchange sites can now offer.

The components in the two dashed boxes of Figure 1 that have been discussed thus far are primarily in support of the transactional model of behavior (the level 2 of earlier Table 1). Although our refinements to the finance box move past level 2, at this point we want to introduce further refinements that move a DSS to level 3. In particular three further framework extensions are needed as are depicted across the base of Figure 1. First, most shopping sites today include or are building a data warehouse that captures and helps to track customer transactions, all user interactions be they registered

customers or casual prospects (the clickstream), and psycho-demographic profile information on customers and prospects that is either volunteered and/or purchased from a variety of sources: e.g., see [8]. This data is analyzed offline and in real-time to build descriptive and predictive models of user shopping and buying patterns (what works, what doesn't work, what trouble is driving them away). From the buyer's perspective, this is the kind of memory one expects from local store owners that recognize their customers preferences and recall the kinds of things they have bought in the past. The second extension to the framework is personalization software (e.g., see [9]) to help provide personalized service that many users want, focusing their search and offering time saving suggestions and value-added content. This works by combining warehouse information with business rules, matching approaches, and/or correlation functions (e.g., collaborative filtering) to help decide what products, product info, and content will be most attractive to and useful for that user. This type of software can effect the items emphasized and displayed in each of the other boxes of the framework in Figure 1.

Buyer decision support need not be entirely software based. A final extension to the framework lies in the ability of users to receive human help during their sessions. At one end of the spectrum, many shopping sites include an email button for submitting problems to a helpdesk person who will respond within a few hours or days. In order to better help users, many other shopping sites now include a 24x7 call center for immediate help in browsing and searching the site, in placing bids or securing financing, and the like. The customer service representatives at the call center might operate as a telephone bank, or they might be given help desk software. This software is called a Customer Relationship Management (CRM) package as it utilizes the results of the warehouse to help the representative manage the overall relationship with and workflow for that customer, not just the single incident or session difficulty associated with the phone call: e.g., see [10].

These three new blocks across the base of the DSS framework raise interesting temporal and scope dimensions that were previously absent from the framework. The Miles and Guttman type framework are based around a single transaction view of the customer. While there is a timeline to such a transaction, it tends to be finite- usually on the order of minutes or hours for the sale, a few days involved in support of users who wish to track the purchase and, a month or so (hopefully) for assuring payment and possible returns and re-crediting. The extensions we include here shift the framework towards an "infinite" timeline, or at least toward lifetime-scale management of prospect and customer information. One can think of a doubling of complexity, where one must mange not only the current shopping experience but also the process of attracting prospects and new customers as well as the customer retention process. The reasons for accepting this added complexity are widely published. For example market analysis [32] indicates that it cost six times more to sell to a new than to an existing customer, and that a firm can boost it's profits 85% by increasing its annual customer retention by only 5%. From this perspective, putting into place customer relationship management capability (help desk) is well worth it since 70% of complaining customers will do business with a company again if it quickly mediates any snafus.

Also only about 2% at most of those who visit a website wind up making a purchase. So keeping that 2% is vital and a key to that retention is the mining for prospects and attracting more customers so there is a bigger pie for that 2% to be drawn

from. In short the marketing theory suggest the extensions to the DSS framework across the base of Figure 1, and we have added them. However there is a large difference between just doing the right thing vs. doing the thing right. This is true of every block in Figure 1, a point we believe warrants significant study as will be further addressed below.

To summarize, Figure 1 provides a framework that overviews the space of design options that a shopping website developer must consider when constructing DSS for the online buyers. Assembling a DSS with all these features is a significant undertaking though it is getting easier. Only a couple of years ago, in 1999, Guttman indicated there were no centralized shopping sites (agents) that also included the right hand side of Figure 1 (bid, ship, finance). Likewise in 1998 when Miles et al researched their framework (published in 1999), natural language was not an option, trouble management was rarely, if ever, offered on the web, gateways and market exchanges had yet to be created, and personalization, warehouses, and CRM software did not exist for web marketplace deployment. Even though their original framework is only a subset of Figure 1's framework, Miles et al. indicated the design point options in their framework significantly exceeded the implementations on the web. Today there are still unexplored design points/options, but there are many hundreds of web-based shopping sites and one can often find an implementation of any given design point or a close approximation to it. Further, there are typically two to four dozen application providers offering pre-packaged solutions for many of the boxes of the Figure 1 framework (e.g., personalization, CRM, search engines, etc.) and a growing number of service sites one can embed that provide other components of Figure 1 (e.g., shipping, finance).

The problem confronting the website DSS developer today is that each of these packages pose a feature-explosion – which ones to choose for their users? This is no small issue and the software package features take nearly as much time to integrate with the online catalogs as it does to build a preliminary version of the feature set from scratch. Also, there is a significant vendor inter-operability problem. Since no application provider yet operates across the full design space of Figure 1, one must seek a mix of software buys and builds that overcome innumerable inter-operability issues. This is leading at least some dot coms to avoid all application providers and try and build the full range of DSS features on their own. Finally, there is a shortage of lessons learned and evaluation literature that can help guide the DSS designer through this maze. What works, what doesn't work, what is reducing user frustration, etc.? How does the website developer sort out the options that the application provider packages offer? Which packages help reduce the problems raised at the outset of this paper and which don't?

This paper offers only a few answers to this large set of design questions. The framework of Figure 1 bounds the space of design questions, and the case study presented in the rest of this paper explores how one group navigated through the framework, their rationale for various design decisions, and some lessons learned. Also in this case study we attempt a two level analysis. On one level we focus on the overall framework without worrying about which design points work best (Sect. 3). After completing that effort we then examine some of the design points with particular attention on search and browse (Sect. 4). As such one can view section 3.0 as verification of our framework for design of level 2 and 3 types of DSS, while Section 4 goes into design points that fall largely within a level 2 DSS. Numerous studies are needed to examine all the design points issues, and we had to choose which ones to focus on first.

Many more case studies are needed before the entire extent of this design space will be well understood.

3) CASE STUDY VALIDATION OF THE DSS DESIGN FRAMEWORK

This case study is of a B2B dot com that has features on their website for each of the boxes of Figure 1's framework. Further, these features are often designed to offer more than one of the options in the boxes of the framework. In this section we overview the website and the current status of its design. Then we turn to an in depth discussion for a select few design features and address the challenges, efforts underway to overcome them, and lessons learned to date.

Specifically, the case study website is EqualFooting.com (www.equalfooting.com), a B2B online marketplace for the "maintenance, repair, and operations" (MRO) sector: e.g., see [29]. This means basically EqualFooting sells industrial and construction supplies – something like a Home Depot for small contractors only with an order of magnitude more products than Home Depot offers. The company's official launch date was February 2000 and by June 2000 they were handling one million hits per day (by about 23,000 separate users daily). Also, at this writing their catalog integrates almost half a million products offered by over 2,000 sellers.

The unified catalog is assembled on an outsourced basis and is then stored internally in an Oracle database on multiple processors to balance user load. A hidden mirror site exists on the opposite side of the country to further address redundancy and load issues. Atop the catalog is a webserver (WebLogic) and a Java implementation of about ½ million lines of code, as of this writing, that performs all the functions of the website and that connect the users' web-browsing clients to the remote servers holding the catalog. At present this involves a three-tier model -- Java Server Pages, webserver, and database -- although it is being evolved into an Enterprise Java Beans architecture later this year.

Figure 2 shows the view of the homepage for EF's website as of this writing. This screen shows 3 main entry points for customers--- search and browse on the left hand, instant finance and credit on the middle and shipping choices on the right. This presentation is different than the typical B2C homepage many readers are probably used to and it highlights the fact that B2B customers have a different set of priorities. Since they typically order larger supplies and need them for meeting their own customers needs, shipping can be a bigger issue than for retail customers. Likewise, instant line of credit and finance is important and transcends just putting the purchase on a credit card. We will discuss each of these panels further in what follows.

The website's tabs across the top Figure 2 support other stages of Figure 1 such as entering the market as a seller, registration and profile (My Account), a transaction history and bookmark capability (Activity Center), Member Benefits, and AboutUs (press room). Also, shown near the upper right are the shopping cart, how to contact the 24x7 customer service (800 number), and various links to background sources. The tabs and upper right links are omnipresent no matter which function the user is performing.

Since launch the company has released a new version of the site every six weeks, and as of this writing is about to release the fourth version. While the color scheme, look and feel, and basic user interface have, so far, remained consistent between releases,

major new capabilities appear with each release. For example, the initial version omitted finance, while the third release included "instant decisioning" for equipment leases and the fourth version includes real-time processing of bank loans, lines of credit, and Small Business Association loans through a variety of partner institutions. The rapidly increasing demand by users for new features such as financing and shipping has lead to revision of the original website business model, and these are now horizontals being marketed in their own right and embedded in other websites.

At launch there was no personalization and no learning about users as shown at the bottom of the framework diagram of earlier Figure 1. The only tracking of customers was through a weekly logfile printout. This included only summary statistics on gross number of users, sellers, transactions, bids, sales, and the like. By mid-year however, a full transactional and clickstream capture capability had been implemented, a star schema for the warehouse had been deployed, standard reports were designed on every aspect of website usage and commercial operation, software for online analytical processing was installed, an appropriate analytical staff was hired, and the warehouse went live. New releases are currently being planned with greater datamining capability, an expanded focus for the warehouse to grow into a marketing tool including prospect information, and not just customer profiles, histories and clickstreams. As of this writing there is still no personalization tool deployed for the website, but personalization solutions and application service providers are currently being evaluated. And launches of initial personalization features are planned. Having the warehouse operational was a precursor to developing the insights needed to successfully deploy personalization features.

Figure 2 – Homepage of the Case Study Website - Equalfooting.com

In terms of non-computerized DSS for the users, a multi- person staff mans the customer support centeorn a 24x7 basis. These "Customer Service Representatives" help by phone, or alternatively in the background, at some point in the processing, of many (if not most) of the bids and transactions since many of the sellers are small businesses with little skill in handling the sudden influx of electronic traffic they have experienced since launch. At launch, the support center was equipped with disparate telephone management, email queuing, electronic guidebooks, and other support systems. Plus they had to logon to the site as the buyer or seller in order to perform transactions on their behalf and/or help them. In the interim they have purchased a Customer Relationship Management (CRM) solution that integrates all these subsystems together, helps manage the overall workflow and unified help queue, provides automated helpdesk capabilities, and provides an independent path into the catalog, including enforced logging of call center representative actions on behalf of users. Integrating the full extent of this "solution" into the website software is taking months of effort by over a dozen individuals, including a number of manually coded extensions and features to get it to work.

In summary this study served as a proof of concept for the DSS framework of Figure 1. One might be tempted to conclude at this point that while this updates the literature on DSS frameworks, that indeed many B2C sites already use the block elements (perhaps with different emphasis on their homepage) and that the literature was lagging the practice. Such statements fail to understand that the DSS approach cannot be successful based upon breadth of coverage alone. Success vs. failure of the DSS and indeed of the overall shopping experience depends on how each block is implemented. For that reason we turn now to an evaluation of principles governing design points within the DSS framework.

4) MIGRATING FROM SEARCH TO DECISION SUPPORT

The prior section illustrated how all the major blocks of the Figure 1 framework are being implemented by the case study website, but what about the finer details of the framework? This is the transition from "doing the right things" to "doing those things right". How are the individual design tradeoffs and choices in any given DSS step made and supported? And, how do the choices in the DSS steps solve the significant challenges raised earlier in the description of product catalogs? As mentioned earlier, examining design points for every block of Figure 1 is a large-scale, long-term undertaking that we have only just begun. In deciding where to begin, we started with high priority area—that which was identified at the outset of this paper as being key to 4 out of the 5 failure modes of shopping sites. That is, this section examines design points concerning decision support for search and browse tasks. These are the boxes of a level 2 type of DSS contained within the inner dashed box of earlier of Figure 1.

The problem is that the available capability from catalog product and application vendors for deploying these features is handicapped by a number of factors that lead to the type of problems cited in the introduction. This is a strong statement that we will support in the ensuing subsections after a few introductory remarks. Specifically, the next

four subsections will provide a critique of currently available search and browse capabilities and try to explain the root causes of the common observed failure modes. We will provide this critique through a mix of techniques including explanation of constraints, description of failure modes, illustrations of critical incidents drawn from log files at the case study site, and comparative performance evaluation.

Rather than present all the difficulties first and then separately address their solutions, we instead interweave a discussion of principles for solving the failure mode right after defining and explaining the failure mode. For each principle we also include illustrations of how to apply it to improve the shopping experience. The goal of this discussion is to allow the reader to gain a sense that these problems can be mitigated, but only by investing effort in their solution. As we proceed the reader will see that one must use every bit of information in the catalogs, plus a wide array of DSS and search techniques to make headway and reduce user frustration. Even with such a wide-scale approach, the problems do not disappear easily, and one must go beyond what the application providers offer.

Before proceeding, it is worth introducing some notation. The most common form of search on the web and in product catalogs is what is called keyword search. This is defined as a match between the t= 1,T terms in the query string (Q_t) and their counterparts somewhere in the j=1,J terms in the kth document (D_{jk}) being searched. Here a document is a record in the product database. If the match is exact, the similarity (Sim) equals unity and the pointer to the document or product is returned.Let us state this as:

$$Sim(Q, D_k) = \{ \sum_{t=1}^{T} Sim(Q_t, D_{jk}) \} / T \qquad (1)$$

Where,

$Sim(Q, D_k)$ = score of the kth document against the query string

$Sim(Q_t, D_{jk})$ = score for the t^{th} term in the query string where score is (0,1) when t and j (don't match, match)

T = number of terms in the search string

A perfect match occurs where $Sim(Q, D_k) = 1$. In general (non-catalog) web searching, most engines will return partial matches often by reducing the threshold to some reasonable number ($Sim(Q, D_k)<1$), but they will sort the documents in descending score so those closest to 1.0 will appear first. Also, when searching on the web, $S(Q_t, D_{jk})$ is most often computed as a weighted dot product of the respective term vectors, Q_t and D_{jk}: e.g., see [29, 30]. However, in catalog search, a strict keyword match is often utilized, where the boolean AND is assumed between all the terms of the vector Q_t. Thus, equation (1) is often utilized in a form like what is shown here. As an example of this conjunctive form of keyword matching, in response to a search for "green chair" most product catalog engines will attempt to find a record with the terms "green "AND" chair", while most websites searching engines will also return every document with "green" or "chair" but these single term hits will appear lower on the list of returned items. Neither of those approaches will find records and documents about "olive seat" or other synonymous terms.

As a final introductory note to this section, the Equal Footing effort postulated that four principles were central. Clearly there are many more than four design principles required for good shopping sites however we have chosen to highlight these four

principles since they attempt to convey the key ideas of buyer behavior theory. We briefly summarize them here and we amplify them as they are utilized in subsequent sections:

- **P1: Use Natural Language Search** – Buyer behavior theory indicates that users will spend substantial time searching and comparing products and makers. To facilitate this effort, just as with a good human sales representative, the search should culminate in what the user wants and not be confounded by how they said it. However current search technologies is almost exclusively keyword matching, which means the given search will only match on exactly what was typed and not what was meant. Solving this dilemma implies use of natural language techniques including synonymizing, spell-checking, and related capabilities to translate the term vector Q_t from the users language into the lexicon of the catalog. To continue the example, the search should find green chairs, olive chairs, chatreuse seats, aquamarine arm chairs, and so on.
- **P2: Build Domain-Specific Search Agents** – The advantage of studying buyer behavior is that one learns the mental process of this class of users is different from that of document searching users on the web. Given such differences one should not try to impose the same type of search engine process on buyers. This includes two connected ideas: the first is that generic search engines won't suffice. The second idea is that once you drop the idea of a generic, one-size-fits-all search engine you need to replace it with an alternative. That alternative should be an agent that understands the domain of buyer behavior and that supports the mental process it connotes. Just as a good store provides sales people in the aisles to help with your selection process (e.g., look for green chairs under kitchen furniture, lawn and garden, and home furnishings), a domain-specific DSS also should operate autonomously to learn the users' goals and to help them accomplish their desires (i.e., an agent able to interpret user $Sim(Q, D_k)$ intent). This implies finding and deploying a domain-specific grammar so that the terms in Q_t may be extracted and parsed to improve retrieval.
- **P3: Treat Search as a Process** –Since the buyer behavior theory focuses on the non-linear, iterative processes of elimination that buyers go through, any new form of search should be supportive of this elimination-by-aspects type of reasoning. The idea is that search might not succeed on the first try, and that similarity finding is a process of narrowing and eventual convergence after several interactions with the user. A goal here is to make the interactions transparent and error forgiving. For example one should readily be able to accumulate green chairs from multiple "departments" Place them side by side for comparison (something you can't do in an actual store), and reach a happy conclusion.
- **P4: Use and Manage Knowledge to Speed Search** – Buyer behavior theory suggest that users do not just blindly search but they use personal taste and preference, criteria, brand, lexicon, and other knowledge to help shorten the search. Users in stores won't walk every aisle to find a specfic item. They expect signs and sales people to help them. Their search on the web should be similarly shortened by this new type of agent. If they search for "green chairs" they shouldn't have to look at "green seat covers for automobiles". This is similar to the old idea in artificial intelligence of adding knowledge to speed and help focus the search. The catalog provides a lot of knowledge about product categories, taxonomies, and ontologies that

should be managed and utilized (the C_M mentioned earlier and the k of the D_{jk}). Likewise a domain specific grammar should utilize meta-knowledge about the structure and representation of products in catalogs (the j in D_{jk} plus the *R3*). Finally, knowledge to help the search process could be culled from every other possible source: e.g., textbooks, the data warehouse, and from users themselves.

4.1) Capturing and Representing the Catalog Taxonomies

As mentioned earlier, the product hierarchy fields hold the multi-level browse tree that one encounters on the web as a set of links from the top of the catalog to the leaf nodes that point to the actual items or products available for sale at that site. The category names and levels in item hierarchies are usually carefully chosen to tell the user the structure of the product offerings, the organization of the catalog, and where to find products. The hierarchy is a taxonomy or ontology of the entire catalog. From the users' perspective, a well-designed hierarchy is often bushy in the middle with many pathways to reach leaf nodes. This is an error-forgiving design so the user's "aim" doesn't have to be very good yet they can still browse down the tree and find the items they're interested in. Thus a user who doesn't know that "grease" is catalogued under "abrasives" will still have a chance to find it.

Unfortunately, as product catalogs grow in size and, particularly, as 1,000s of different catalogs are merged (each with their own unique hierarchy), it becomes increasingly manpower intensive and difficult to establish a common taxonomic structure for a marketplace. The individual manufacturers and suppliers generally can't afford the extra manpower to conform to market makers' categorization scheme. A common compromise is that the market makers create the upper layers of the hierarchy so that it conforms to their concerns for browsability, understandability, and error forgiveness. At the lower layers of the hierarchy, they often meld in the sub-hierarchies of different suppliers/manufacturers where they exist and create branches where needed. The result is a pragmatic compromise, though it lacks the rigor of an industry-approved standard.

One alternative is for each industrial sector to adopt a standard for naming and taxonomic indexing of items. In the field of book publishing there is the Dewey Decimal System and all new books (each with unique ISBN numbers) are categorized within that taxonomy before being sold. Within other industries, however, the world is rarely so standardized. Still principle P4 is important, and one can find groups working on it.

Several attempts to provide standard taxonomies such as the United Nations' SPSC commodity codes [23] or the US Government Supply Agency's "National Stock Numbers" are promising in the mid-term but are insufficiently developed at this date. For example, the UN/SPSC only provided 78% coverage of the over 13,000 categories of products in Equalfooting.com's maintenance and repair item catalog. To adopt it, one would have to petition the standards setting group to extend the taxonomy to cover all the products in a given catalog, a catalog that is continually and rapidly growing. Further, the UN taxonomy introduces non-intuitive category names (e.g., nuclear fuel rods are indexed under "lubricating preparations"); limits itself to 5 levels of categories even though it covers the entire world economy (this is insufficient depth to provide adequate distinction between lower level taxonomic categories in a given industrial sector); and imposes strict tree structures so that leaf nodes may have only a single pathway to them.

This makes them unappealing for browsing purposes. Similar pragmatic concerns also currently exist with the category hierarchy for the US' National Stock Numbers.

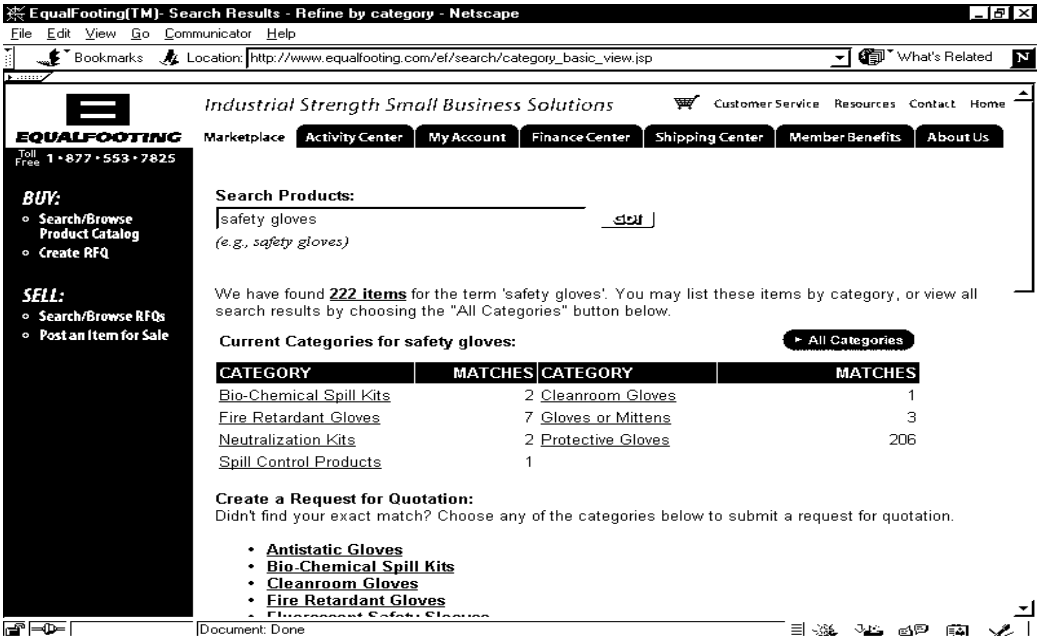
The market-maker generated category hierarchies provide a rich resource for figuring out how to overcome the concerns with the evolving product hierarchy standards. In the future, one can expect that communications will occur between these groups and that industrial sector standards will evolve into taxonomies useful for e-commerce. In the interim, the market-makers' category trees serve as the best source of product location information that browsers have and that search engines could and should make use of.

4.2.) Integrating Browse and Search

Research shows that about 50% of the users are search-dominant, about 20% are browser-dominant, and the rest use mixed strategies [3]. This implies both the browsing catalog and search engine box should be on almost every page the user sees. Further, the mixed strategy users may want to be able to search within a category they have browsed to. Most engines reviewed in the next section can be programmed to permit such usage but that is not the default capability. Websites need to provide a button for users to toggle so they can perform "within category" search, rather than the across full catalog default option that search generally utilizes. This button may not be found by most users, but some will eventually grow to notice it. Different from this is the within-category sort, usually by one or more parameters – a topic we address in Section 4.4.

There are a number of guidelines for catalog browsing mentioned in the literature that are good to follow, although, we won't be able to provide the space to look at all of these here. A few worth mentioning are to: (1) display picture icons with each item in the catalog, (2) make sure the "buy" and "bid" buttons are available in each view of a product (single liner, high level, full detail) so as not to interrupt user buying decisions, and (3) bring search engine users to a category table before taking them to the lengthy list of hits of their query. This last item slows their search, but injects a screen that reduces the kind of experience mentioned in the opening quote of this paper. Figure 3 provides an example of such a screen for a search on "safety gloves". Here we see that rather than immediately sending 626 hits to the screen, the user encounters an intermediate page that lists all the categories that hold different types of safety gloves, along with the number of gloves included in that category (Principle P3). This helps them better focus their search (Principle P4), and simultaneously learn a bit about the catalog. If they toggle a category, they will see a listing of all the gloves in that category with a picture icon plus buy/bid icons next to each line of the listing. Alternatively, the bottom of the page on Figure 3 links them directly to placing low cost bids (request for quotations) for gloves, including the ability to describe the exact kind of glove they want and how much they are willing to pay for it, thereby bypassing the need to do any more search or browsing at all.

Figure 3 – An Example of How Integrating Browsing With Search Reduces User Frustration About Receiving 626 Hits They Don't Know How to Sort Through.



An examination of the Equalfooting clickstream shows that some of the more adept users logon to the site and very quickly come to this type of page, briefly drilling into a given catalog item to inspect some of its attributes, and then immediately request a quotation (ask sellers to give them low cost bids). They tend to publish several such requests in the space of a few minutes of search and browse activity. The principles and guidelines just mentioned seem quite suitable for helping them.

In contrast, the more common, and less adept users logon and have a far more frustrating experience. The search button logfile shows innumerable examples of people searching the identical item multiple times like this person who repeats the search 4 times for "hard hats" (note: all subjects discussed in this article are anonymous users drawn from the transaction logs and clickstreams)

Date	Search Type	Search String	Matches
05/04/2000 01:36:33	Keyword	hard hats	213
05/04/2000 01:38:34	Keyword	hard hats	213
05/04/2000 01:39:07	Keyword	hard hats	213
05/04/2000 01:39:40	Keyword	hard hats	213

This is an example search for a broad item (e.g., like safety gloves), then drilling down through the category table to peruse the choices. Presumably the users don't know how to use the BACK button on their browser (which would be much faster than re-searching EF's catalog), so they hit GO again and repeat the slower search so they can return to the category page (Figure 3) and explore categories they didn't drill into before. Our primary lesson learned here was that most users aren't adept at using their browser, and that the best way to support them is to allow them to effortlessly search and browse as they wish. To better support this pattern of searching, we made it a priority to increase the search engine speed by an order of magnitude, a goal we achieved in mid-2000 by upgrading from Oracle 8.1 to 8i and from a few internal tuning changes.

To date, we still haven't been able to explain the behavior of a few users such as the following one who reached the category table and even though it indicates 0 hits, they repeat their identical search, sometimes multiple times. However, given what product this category of user often seems to be looking for, we're not sure we want them lurking about our website anyway:

Date	Search Type	Search String	Matches
05/11/2000 09:15:13	Keyword	GUILLOTINE	No Matches
05/11/2000 09:16:33	Keyword	GUILLOTINE	No Matches

4.3) Improving Keyword Search of Product Description Fields: The Munge

To date most catalog search engines do not use the field(s) holding item category hierarchies to help guide their search for products. As mentioned in Section 1.3, they rely heavily on a second relation that holds the item description (document fragments), what we defined earlier as $R2 = \langle \mathbf{D_1}, \ldots, \mathbf{D_K} \rangle$. This is a free text field that suppliers and manufacturers generate and use in their hard copy catalogs. Its contents (coverage, quality, and clarity) are at the discretion of the individual vendors. Often, these fields hold the suppliers' insights about what information the buyers will want to see. However, there may be little consistency between different suppliers' conceptualizations of buyer needs, so the descriptions will be difficult to compare.

The market maker's role is to assure that these descriptions get captured into the product database and used by the search engine. For example, at Equalfooting.com the catalog includes two description fields – the so-called short and long description -- although, many suppliers omit the long description. An example of a short description is '2" PURE BRISTLE PAINT BRUSH', while the corresponding long description for that item is '2" Professional quality MASTER PAINTER brush. 100% pure black china bristle. Handcrafted in the U.S.A. This is a $10.00 retail brush that we are closing out while they last.' Interestingly, while the e-catalog includes a field called "item name" the suppliers have no such data to provide for that field. They use the short description in place of a name in their hard copy catalogs. The category tree "bread crumb trail" to browse down to this item in the e-catalog provides significant information about what it's name ought to be though: Product Catalog > Industrial Supplies > Painting Supplies > Paint Brushes. So one could get the search engine to use (leaf node) category names as surrogates for item names. Although this is not generally the practice elsewhere, EqualFooting does add this leaf node into a field called the "munge" that holds all descriptions. This leaf node name is one more piece of knowledge about what the item is (Principle P4) and it helps the search process. Also added to this munge are the full set of product attributes and value fields (as further free text entries) so that the search engine can try and match parameter queries against these items directly. Thus, the D_k used here combine information from *R1*, *R2*, and *R3* to try and make the D_k into more robust document fragments.

In general, several challenges confront one attempting to support search of item description fields, a few of which the munge is an attempt to deflect. First, is the question of the completeness and consistency of the product descriptions across sellers. Some descriptions include partial parameter information (e.g., 2", black) while others focus on how the product might be used, and still others are highly terse and omit most details.

Second is the difficulty of matching the buyer's search terms to the wording in the descriptive fields. Where exact term matches don't exist, one must consider issues related to word stemming, spelling errors, abbreviations, synonyms, and related problems. A third challenge is that numeric attributes are poorly and incompletely represented in these descriptions, yet some users will want to search by size, weight, height, voltage, and many other quantitative parameters. Unfortunately, the item description fields do not permit structured search of attribute information. For example, if searching the item description field for 2" brushes, a keyword search engine doesn't know this is a width attribute and it would return hits on any occurrence of 2 and brushes. Thus the engine will return 2 ½ inch brushes, brushes made at 2 Downing Lane, and so on. Finally, there is the challenge of trying to figure out the user's intent and underlying search goal. We address these challenges and some solutions to them in Sections 3.4 and 3.5.

4.4) Trouble Management Design Issues: Complementing Keyword Search

Earlier we mentioned that most users encounter search failure and frustration due to sites having poor DSS and being unable to properly support them. Research shows that about 15% of online search failures are due to spelling errors and another 40% are due to customers using different terms from those in the website (e.g., patching vs. concrete): e.g., see [1-3, 13]. Because they can't interpret the meaning of the users' query, search engines typically are tuned to bring back innumerable hits of everything even remotely relevant, often burying the best choice deep within the list (or omitting it altogether) and providing little help for then searching just the returned set of items. In short, when users get in trouble, most shopping sites provide little in the way of "trouble management".

Some usability engineers [2, 3] argue that search engines should aim to communicate the concept that searching on the Internet is a process rather than an event. It is relatively rare to find something useful as a result of a first search. An important design goal is therefore to engineer this view of search as a process into the interface itself. This has obvious implications for the wording of the instructions used, for example expressing search results as "suggestions" rather than "hits". This is Principle P3.

A recent example at our website prior to our trouble management approach lead to the following logfile that illustrates a customer who's a weak speller, but fortunately he understands this principle and is a persistent searcher.

Date	Search Type	Search String	Matches
05/15/2000 12:14:33	Keyword	cobbelstone	No Matches
05/15/2000 12:15:04	Keyword	drivway	No Matches
05/15/2000 12:15:20	Keyword	driveway	1
05/15/2000 12:15:43	Keyword	asfalt	No Matches
05/15/2000 12:19:28	Keyword	stone	73
05/15/2000 12:19:58	Keyword	curb	14
05/15/2000 12:21:45	Keyword	stone	73

Many other users are not nearly so patient and persistent. Shopping sites shouldn't have to rely on subtle messages, and user patience alone. They also need to offer improved DSS capability that anticipates error, helps with corrective suggestions, and supports the user at their current skill level – an error forgiving approach.

However, most shopping websites use keyword searching engines that are ill-prepared for trouble management. These engines can ignore upper and lower case letters, and retrieve hits regardless of capitalization, but they aren't so good on more serious error corrections. Few of them correct spelling errors. Also, keyword searches have a tough time distinguishing sense -- words that are spelled the same way, but mean something different (i.e. hard cider, a hard stone, a hard exam, and the hard drive on a computer). This often results in hits that are completely irrelevant to the query. Some search engines also have trouble with so-called stemming--i.e., it should recognize the word "big" also implies a hit on the word "bigger"; it should equate singular and plural words (e.g., work, works); and recognize alternate verb tenses that differ from the word entered by only an "ing," or an "ed". Search engines also cannot return hits on keywords that mean the same, but are not actually entered in the query. A query on heart disease would not return a document that used the word "cardiac" instead of "heart." That requires expansion of the search to include synonymous terms. Finally, strip words alternatively often need to be removed from a search string or are missing from a title that can cause an engine to fail to notice a match – e.g., articles like "a," "an," and "the."

Unlike keyword search systems, natural language and concept-based search systems use all these capabilities (spelling, sense, stemming, synonyms, stripping, etc.) to try and determine what you mean, not just what you say. In the best circumstances, a conceptual search returns hits on documents that are "about" the subject/theme being explored, even if the words in the document don't precisely match the words entered into the query (this is Principle P1). Fortunately, many commercially available search engines today have conceptual based searching features that help one to manage trouble and user error. Table 2 shows some of our efforts to compare and contrast five such engines. These engines include two very robust, full-featured systems -- Oracle's Intermedia, (free to those using Oracle as the database for their catalog) and AltaVista's search engine (used by Amazon): e.g., see [14, 15]. In addition are three intriguing specialty engines including EasyAsk [16] which provides small-scale, tightly focused catalogs with very powerful conceptual searching; Frictionless [17] which offers a decision theory model for parameter and attribute search (used by Lycos Shop); and one we call Brand X (due to non-disclosure agreement) which attempts to use probabilistic match to manage trouble.

Table 2 shows that one can support the integrated browse and search ideas mentioned earlier by programming an interface to any of these engines, although EasyAsk has some of this already built in. In terms of keyword search, we were able to install three of the engines atop a version of our 250,000-item catalog, so they could search the "short and long description" fields. We then tested these engines on a range of keyword searches, a few of which are in Table 2. First we tried single word tests three of which are shown. The catalog has 226 actual hammers, although many other items are related like jackhammer, hammer drills, hammer chisel bits, and the like. None of the engines agree on the number of hammers Also, as explained earlier, we have Oracle set up to search a field that merges all other catalog fields into one (the "munge") so it picks up 805 extra hits on an electronic parts manufacturer whose name is C-H CUTLER HAMMER Inc. In terms of the 'cord' and '20-amps' searches, we think Oracle is the closest to what is actually in the catalog. With the "cord" search, we can't explain the differences since this term only appears in description fields, although with "20-amps" we suspect the variations of the other two engines are that Oracle could also search the

attributes (which are dumped into the munge), not just the descriptions. So this is not a fair test of the other engines.

Table 2 – Comparative Overview of Feature Sets Offered By Illustrative Concept Based Search Engines

Criterion		**Oracle** 8i **InterMedia**	**Alta Vista SE 3.0**	**Brand X**	**Easy Ask**	**Friction less**
Search Methods		Keyword & Concept	Keyword & Concept	Probabilistic Match	Concept	Decision Theory
Integrated Browse & Search		API call	API call	API call	Built In	API call
Keyword Search:						
Single Word Trials	Hammer Cord 20-amps	1,512 hits 480 hits 679 hits	605 hits 930 hits 7 hits	480 hits 2 hits 0 hits	-- -- --	-- -- --
Multiply-Worded ItemTrials	Electric Chain Saw Bolt Cutter	1 (4) hits 30 hits	0(1) hits 35 hits	427 (466)hits 593 hits	-- --	-- --
Conceptual Search:						
Stemming		Built in	Built in	Built in	Built in	NA
Strip Word Removal		Built in		API call		NA
Spell Checking	pwer powr hamer recepticel	-- -- -- --	Per Power Hammer --	-- -- hammer receptacle	Built in	NA
Synonym Table		Extendable	Extendable	API call	Extendable	NA
Strip Word Removal		API call	API call	API call	API call	NA
Parameter Search:						
UserPrefs		NA	NA	NA	NA	Built in
Parameter	Single sort Multi-sort	API call	API call	API call	API call	Built in
Compare		Programmable	Programmable	Programmable	Programmable	Built in
Miscellaneous:						
Catalog Crawler		Built in	Built in	API call	Built in	Built in

In terms of the multiple word searches, Oracle found the correct number of hits for “electric chain saw” and “bolt cutter”, while AltaVista was close (though disturbingly made key misses). Numbers in parentheses are for the search on just the words: “chain saw”. These engines use AND search and only return items that have all words of the search. Brand X, on the other hand, exclusively uses a probabilistic matching algorithm that retrieves hits for each word in the phrase, supposedly placing hits that have all three words higher on the list. Oracle can support probabilistic matching, but we did not turn it

on for this test. The lessons of this comparison exercise are that one needs a full-featured search engine so as to avoid getting trapped into a single approach, like probabilistic match (or decision theory). It is important to be able to tune the engine and turn on and off various features. In this regard, both full featured engines appear about equally robust: Oracle and AltaVista.

In terms of the conceptual-based searching features, most of the engines reviewed offer a relatively full complement of these features. Oracle omits any spelling checker, however, Java source code for a robust spelling checker that works as well as what we tested in any of the others can be purchased from the web for under $1,000: e.g., see [18]. In terms of parameter search, only Frictionless includes a robust version of this out of the box. The other engines could be made to support this, but significant programming and API calls are needed. In the next few sections we explore what is needed to install and activate conceptual-based search and parameter search. The reader will see that there is no panacea and a lot of effort is needed not only to program the features, but also to research and decide how best to deploy them. In fact, adding these capabilities atop one of the search engines is the first step of introducing a new querying engine paradigm: e.g., see [21, 22] – what P2 calls a domain specific search agent.

4.4.1) Obstacles to Setting Up Natural Language Search (P1)

The natural language and/or conceptual search features are not the default settings, are difficult to set up and operate, and require significant investment and effort before being useful in any given site. Further, not all engines offer the same features or provide the same degree of trouble management capability. As an example, to adapt Oracle Intermedia to support EqualFooting, has required about a man-year of effort so far. One task was to design the agent algorithm as the next section describes in more detail. Another task was to integrate that agent with the Oracle search engine and with the catalog. A third task was to create the three dictionaries always needed for conceptual search – spelling, stripping, and synonyms – as these are domain-specific items. Although it came with 100,000 words it was necessary to embellish the spell checker's dictionary by adding: (1) the top 1,000 mis-spelled words from the user search log and their corrections, (2) proper names of all manufacturers and suppliers (the spell-checker assumes initially that all proper names are errors) and how they might be mis-spelled, and (3) many dozens of acronyms with proper spelling (e.g., CD, DVD, HVAC, etc.).

Likewise, Oracle Intermedia includes a synonym processor, but provides no thesaurus. Growing this thesaurus involved many false starts and deadends. For example, it is tempting to try and use an existing general purpose thesaurus such as WordNet from Princeton. This includes 95,000 words and all their synonyms, however, this thesaurus brings back too many synonyms, many of which are inappropriate (racial slurs, curses, body parts, religious terms, etc.). Plus most of the specialty terms of the domain are omitted (e.g., chain saw, Phillips head, and safety gloves). An alternative was to extract all the unique terms from the catalog and to manually create synonyms for these, however, this rapidly exploded into too large a task. In the end, a thesaurus was incrementally grown from studies of the user search logs (initial search string vs. eventually successful string), and from having the call center author synonyms for the 1,000 most frequently non-matched terms that actually exist in the catalog under a

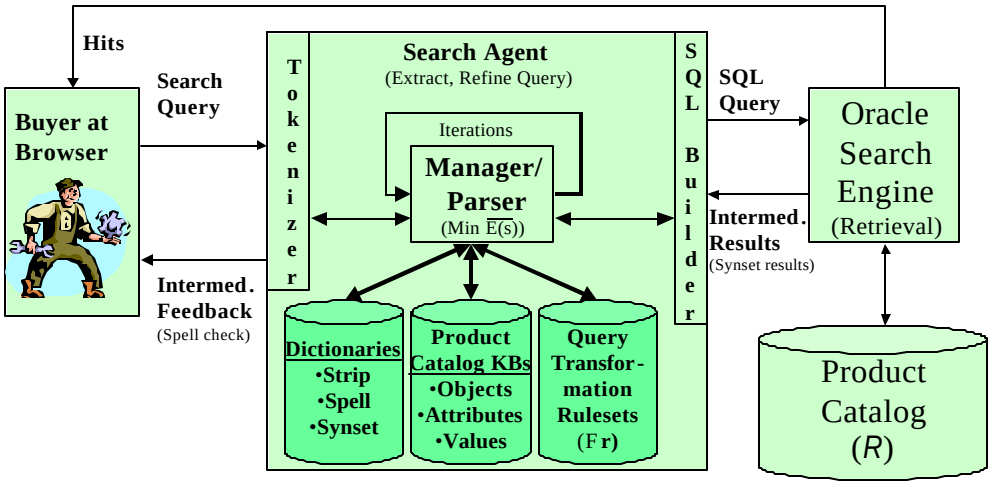
different name. On average, about 3 synonyms were created per term. Also, we added synonyms for each of the multi-term descriptors of the 13,000 nodes of the catalog's browse tree by creating altered spellings of these (e.g., smashed out the spaces, added hyphens instead of spaces, and created single word equivalents).

4.4.2) Domain-Specific, Natural Language Agent (P2)

This article began by mentioning how e-commerce executives have ignored the search process, relegating it to off-the-shelf technology that doesn't work out. Subsequent sections explained how the components of a robust, conceptual-based, and trouble management search approach can be purchased, and then adapted with domain specific knowledge such as the additions to the dictionary and thesaurus mentioned above. In this section, we come full circuit to the notion that one must build a domain-specific search agent that sits between the user and the domain-independent search technologies in order to pull all the pieces together and to support the buyer decision making and trouble avoiding needs of the users.

But building such an agent need not be an overwhelming task. First of all, as just mentioned, such an agent sits between the user and existing components – so half of the task is taken care of just by calling up these components at the appropriate time within the decision support framework. The other half of the task is dispensed with by instantiating the domain knowledge elements of a rule-sequence phrase parser as commonly used in the information extraction community: e.g., see [21, 24, 26]. This is the old idea of rule-based inferencing, applied to the information extraction task. Figure 4 overviews this agent and we explain it further below.

Figure 4 – Overview of Search Agent Architecture & Query Extract/Refine Process



Specifically, a rule-sequence phrase parser takes an initial search string or phrase, and then iteratively labels and subsequently modifies these labels by applying a sequence of transformation rules which attempt to reduce the remaining ambiguities and residual errors left in place by the previous rules. To facilitate this transformation process we first tokenizing the terms in the search phrase, Q. Next we eliminate stop or strip words ("find me a", "get all", and opening and closing quotes) by comparing the search string to a

negative dictionary. Finally, a stemming algorithm must be applied such as in Porter [27]. The result is a new version of the query we label as Q'. A similar process is performed on the document base resulting in D'.

One can view the rule-sequence processor as a finite state automaton that minimizes mean error for each term in Q' across the range of possible states that that term can assume. States for a given term, t, are label settings determined by the various transformation rules (unlabelled token, object, attribute, etc.). More precisely, we may state the objective function in a mathematical (or dynamic) programming formalism as:

$$\left.\begin{array}{l} \text{Min } Z = \overline{E(s)} \\ (s=1,S) \\ \\ \text{Subject to,} \\ \\ \overline{E(s)} = \{ \sum_{t=1}^{T} e_t(s) \} / I \\ \\ e_t(s) = \{ \text{Sim}(Q_{ts}', D_{jk}') \} - \{ \text{Sim}(Q_{ts-1}', D_{jk}') \} \\ Q_{ts}' = \Phi_r(Q_t') \\ \text{Stopping rule: } e_t < \beta \end{array}\right\} \quad (2)$$

Where,

Z = objective function to minimize the mean error across states, $\overline{E(s)}$

t = number of terms in the search string, as in Eq. (1)

s = current state counter for the query string

s-1 = prior state before latest rule transformations

Sim (Q_{ts}', D_{jk}') = score of the t^{th} query token against the j=1,J terms of the kth document. Score is (0,1) depending on whether a match occurs.

Φ_r = application of one of a number of possible transformation rules that changes the state, s, of the query string by tagging or labeling some terms within it.

β = threshold

One need only come up with a reasonable sequence of transformation rules (r =1,R) to utilize this approach. As an example transformation rule, mis-spelled words must be flagged and iteration included to the user requesting them to accept the suggested or alternate corrections, or to add their own. In product catalog domains, another set of transformation sequences that leap out is the concept that searches are usually for objects with attributes of a certain value. Thus many searches are for various orderings of OAV triplets ('mini sized amps', 'hammer colored red'), or some variant where the O is sought initially and then the AVs are subsequently used for comparison and search refinement (bolt cutter followed by size, price, and availability). Even more common are searches for one or more V of a given type of O where the A is suppressed (AA Eveready batteries, ½" no.8 slotted screws, desk chair). In shopping domains, OAV triplets can be thought of as a parsing sub-grammar. In fact a sub-grammar can be derived for objects and another for attribute-value pairs. By crawling the catalog and extracting all unique terms along with indexes to their locations, one can readily construct these sub-grammars. The

resulting KB lookup tables can be utilized by transformation rules of the phrase parser to infer and insert O, A, and V labels onto the various terms of the search phrase. Where labeling isn't immediately possible, still another transformation rule might attempt to expand the query terms via synonyms and then repeat the OAV lookup table process.

Finally, one can add all kinds of domain-specific rules for any given catalog. In the hardware domain some of these pertain to recognizing and labeling the notation. For example, a number followed by ' is feet or " is inches, while a number followed by # is probably a weight. Likewise, the term "made by" is a two value predicate often preceded by an object and superseded by a manufacturer (so C-H Cutler Hammer or just Hammer can be discerned as the object or the maker). Or other rules might be added about two measures linked together, like ½ x 8 for screws. One of the advantages of the rule-based approach is that rules can be added on the fly, and as more get added, the better the search engine becomes. Thus one can peruse sources such as [28] and manually extract rules over time, or one can try to deploy machine learning approaches and add newly discovered patterns as convenient.

Using this approach EF has been able to demonstrate improvements in search response. Table 3 summarizes some illustrative test results where the first column shows the feature being tested, the search string used in a given test is in the second column, and the last two columns show the hits with the keyword (munge) vs. agent search approach. The base case tests of the first set of rows show that the agent recognizes terms within the OAV framework. Thus it finds 'cords' missed by the munge due to synonyms, avoids confusing hammer as a maker rather than just as a tool, and is not stymied by the strict spelling of chainsaw without a space between chain and saw (another thesaurus success). Likewise the agent's spell checker is successfully demonstrated in the second set of rows, while the third set of rows illustrates a couple of tests of the stripping and stemming transformations. In both these sets of rows, the keyword search of the munge comes up with no hits whatsoever. In the fourth set of tests, the keyword search of the munge produces either far too many hits (e.g., finding boxes of 20 fuses that are 30 amps each, etc.) or none, while the grammar is able to recognize the attribute-value pairings correctly in both tests. Finally, where the attribute is omitted in the final set of rows, this seems to help the keyword approach come closer to the Agent. However, closer inspection of the results from the keyword search show it is retrieving irrelevant items such as products that need an AA battery in order to be operated, white gloves made by Red Devil, and so on. The items retrieved by the agent, on the other hand, are all AA batteries and red gloves actually for sale in the catalog.

These results are promising, and EF is currently attempting to scale up the agent search approach to handle the full subgrammar of the 201,000 attribute-value pairs and the 18,000 categories of objects in its catalog. Also, work is underway to add more domain rules, fully integrate all the components, exhaustively test the performance, and refine it where needed. Plans call for all this to be completed and for it to be moved to the production website before the end of the calendar year. However, the approach is modular and parts may be deployed as they mature. For example, the synonym list and stemming are already in use at the production website, while stripping and spell checking will soon be activated. Also the approach works in tandem with the keyword search. Where the agent can't parse a search string, the keyword approach takes over. Thus an answer is always found if one exists.

Table 3 – Illustrative Results from Initial Tests of the Domain-Specific Agent

Trouble Management Feature Being Tested	Search String Used in Test	Keyword Search of Munge	Agent Search
Base case: single word search	cord	480	544
	hammer	1,512	378
	chainsaw	0	2
Rule Transforms: Spell checking	hamer	0	378
	teh chansaw	0	2
Rule Transforms: Stripping stemming, object recognition	List the cords	0	544
	Find me all hammers	0	378
Rule Transforms: OAV triplet recognition	20 amp fuses	495	59
	30,000 btu air conditioner	0	1
Rule Transforms: Value and Object recognition	Aa batteries	33	10
	red gloves	13	5

5) CONCLUSIONS AND NEXT STEPS

This article offers a framework and guidance for developing decision support and search technology appropriate for aiding buyer behavior at e-commerce catalog sites. We presented and evaluated the full DSS framework in broad-brush and then provided fine-grained research on design points for the search and browse portions of the overall framework. In future research we hope to extend our fine-grained analysis to the remaining blocks of the full framework, but we began with browse and search since it is so central to a majority of the failure modes of the online shopping experience. The lessons learned in this research can be summarized as follows:

Do The Right Thing: The decision support framework points out that buyers seek to refine their decision criteria as they uncover and compare products. They must do this untroubled by a morass of challenges that online catalogs pose, some of which are: missing ontologic and taxonomic standards, inconsistent terminology across sellers in a given market, separation of category from product descriptor fields, lack of product name information in the catalog, incompatible use of terms in the catalog vs. user-chosen term, missing data, attribute or parameter names exist as data rather than as field names, and so on. These pose problems for search technology that few engines currently on the market are able to overcome. Further, the framework also points out that online buyers need more than just product search, and when deciding on purchases they also seek to perform comparative studies, engage in low cost bidding activity, and obtain shipping, financing, and availability information. In addition, they seem to desire stores that remember their preferences and offer personalized service, though they aren't willing to spend much time divulging private information, and they resent sites that get it wrong.

Supported by a case study analysis, this article amply demonstrates that no application providers in this industry currently support the range of features one needs to deploy to help manage the trouble that arises when buyers search catalogs. E-Commerce websites currently must make up for this missing industrial capability by taking

responsibility for design, programming, integrating, testing, and rolling out of the needed feature sets on their own. Often one can utilize the help of multiple off-the-shelf components, but these pose significant integration and feature extension needs.

Do The Thing Right: Derived from theory and supported by a real world case study, the DSS framework helps readers to understand that designing search and decision support for e-commerce is a very different issue than designing it for information retrieval tasks on the web. Yet few B2B e-commerce website executives currently seem to be aware of this difference, and too often they attempt to use off-the-shelf web searching technology where it doesn't apply. Even in B2C shopping sites where they seem to understand the framework in general terms (i.e., at the block level), closer examination of the design points shows these sites still suffer many of the same failure modes as B2B sites.

This article also presents four principles to guide design of the needed feature sets, focusing on minimizing the failure modes of the browse and search subset of the overall DSS framework as a starting point example (in the future we hope to research principles for other portions of the framework): use natural language search (P1), build domain-specific agents (P2), treat search as a process (P3), and use and manage knowledge to facilitate search (P4). The second half of the case study explores how these principles guided the development of an improved decision support capability. This improved capability was developed with the help of off-the-shelf technology for conceptual based search. However, that technology did not work out of the box, plus it only supported portions of the buyer decision support framework proposed here. Effort was needed to embellish this technology by adding significant website screen functionality; by crawling the catalog to extract, index, and grow a domain-specific thesaurus, dictionary, and catalog knowledge bases; and by inserting an intelligent agent between the user and the search engine to support buyer needs. This agent is based on transformational rules that one can add incrementally to increasingly complement and supplement keyword-based search. The case study serves as an example of what was needed at this site, though we believe the principles and lessons learned could be generalized to other sites as well.

There is no intent here to suggest that "one size fits all" in terms of a solution to buyer decision support needs. The framework attempts to point out the space of functionality one must consider when assembling decision support for online buyers. The approach in the case study was one path through all the boxes of the framework. Further indepth case studies would be useful to help fully define the framework. In the interim, however, the principles followed here and approach pursued in the case study can serve as a point of departure for other e-commerce catalog sites that are seeking guidance.

In terms of generalizations, one other thing the case study does point out is that industrial application providers still need to go a long distance before they can offer e-commerce websites the range of support that is essential to their survival. Vendor after vendor show up at each online catalog executive's doorstep claiming to have all the features needed, often claiming that the expense of their solution (usually deep into six figures just to get started) and the list of other clients who bought it is proof that they're right. The website executives have little basis of comparison and few defenses against this onslaught, given how distracted they are by other demands of startup. The framework and lessons learned offered here, however, serve to point out that easy solutions do not

exist. At present, one cannot buy the needed functionality off-the-shelf, the industrial application providers just aren't at that stage as yet. The current state of the practice requires that off-the-shelf components be substantially extended and that one must plan for significant integration and domain-specific development effort. This problem is endemic and it won't fade away quickly. Website executives beware! If you want customer-keeping technology, be prepared to assemble from parts that aren't all there yet.

REFERENCES

1. AltaVista Search Engine (AVSE) version 3.1, available from www.altavista.com
2. Angwin, J. (July 10, 2000). (front section) "EqualFooting Proves Funding Is Easy, For B-to-Bs With Something to Sell," *Wall Street Journal* – Interactive Edition.
3. Anon, (2000a)."Points eCRM Architecture: Providing Future Technology Today," avail. from www.teampoint.com.
4. Anon, (March 2000b). "Winning the Online Consumer: Insights Into Consumer Behavior," Cambridge: Boston Consulting Group, www.bcg.com.
5. Anon, (1999). "Why Most B-To-B Sites Fail," Cambridge: Forrester Research, Dec. (www.forrester.com)
6. Anon, (March 2000c). "Revolutionizing the Search for Products at e-Commerce Sites," Littleton: Easy Ask Inc, www.easyask.com.
7. Anon, (2000d)."Datasheets on Frictionless Search Engine," avail. at http://www.frictionless.com/solutions/datasheetform.html
8. Anon, (2000e). "Using the UN/SPSC: Why Coding and Classifying Products is Critical to Success in Electronic Commerce," Granada Research, (avail. from www.unspsc.org/h_using.htm)
9. Anon, (1998). Oracle 8 ConText Cartridge: Workbench Users Guide, Redwood City: Oracle.
10. Aridor, Y., Carmel, D. Lempel, R. et al. (July 2000). "Knowledge Agents on the Web," in Cooperating Intelligent Agents Workshop Proceedings, Boston: ICMAS.
11. Beckwith, R., Miller. GA. Tengi, R. "Design and Implementation of the WordNet
12. LexicalDatabase and Searching Software," http://www.cogsci.princeton.edu/~wn/
13. Cohen, WW. (1998). "A Web-based Information System that Reasons with Structured Collections of Text," Autonomous Agents Conf. Proc., pp. 400-407.
14. Ettlinger, S. (1998). The Complete Illustrated Guide to Everything Sold in Hardware Stores, New york: Macmillan.
15. Faloutsos, C., Oard, D. (1995). "A Survey of Information Retrieval and Filtering Methods," avail. As UMIACS-TR-95-33, College Park: U of MD.
16. Guttman, RH., Moukas, AG., Maes, P. (1999). "Agent-Mediated Electronic Commerce: A Survey," in M.Klusch (ed), Intelligent Information Agents, Berlin: Springer, avail. from http://ecommerce.media.mit.edu. (Also, a version of this paper exists as Maes, P, Guttman, R, and Moukas, A, "The Role of Agents as Mediators in Electronic Commerce." Special Issue of Knowledge Engineering Review on Practical Applications of Agents, summer 1998).

17. Hagen, PR., Manning, H., Paul, Y. (June 2000). "Must Search Stink?", Cambridge: Forrester Research. (www.forrester.com)
18. http://www.kdnuggets.com/
19. Jonker, CM. (July 2000). "ICEBERG: Context mappings," in Cooperating Intelligent Agents Workshop Proceedings, Boston: ICMAS. July 2000.
20. Keen G. W. P, Morton S. M., (1978). Decision Support Systems: An Organizational Perspective , Reading Addison-Wesley
21. Kalakota, R., Robinson, M. (1999). e-Business: Roadmap for Success, Reading: Addison-Wesley.
22. Kaplan, S., Sawhney, M. (May-June 2000). "E-Hubs: The New B2B Marketplaces," *Harvard Business Review* pp. 97-103.
23. Miles, GE., Howes, A. Davies, A. (2000). "A Framework for Understanding Human Factors in Web Based Electronic Commerce," in *Int. J. Human-Computer Studies*, v. 52, 2000, pp. 131-163.
24. Neilsen, J. (July 15, 1997). "Search and you *may* find" Alert Box, avail. at http://www.useit.com/alertbox/9707b.html
25. O'Keefe, RM., McEachern, T. (1998). "Web Based Customer Decision Support Systems, Commun. Of the ACM, v.41, pp. 71-78.
26. Pazienza, MT (ed.), (1999). Information Extraction: Towards Scalable, Adaptable Systems, New York: Springer.
27. Peppers, D., Rogers, M. (1999). Enterprise One to One: Tools for Competing in the Interactive Age, New York: Doubleday.
28. Pollock, A., Hockley, A. (March 1997). "What's Wrong with Internet Searching," *D-LibMagazine*, Ipswich, UK: BT Laboratories. avail. at www.dlib.org/dlib/march97/bt/03pollock.html
29. Porter, MF. "An Algorithm for Suffix Stripping," avail. with coded implementations at www.muscat.co.uk/~martin/def.txt.
30. Sentry Spelling Checker avail at http://www.wintertree-software.com/dev/ssce/java/index.html
31. Silverman, BG. (1992). Critiquing Human Error: A Knowledge Based Human-Computer Collaborative Approach, London: Academic Press.
32. Vilain, M. (1998). "Inferential Information Extraction," in MT Pazienza (ed.) Information Extraction: Towards Scalable, Adaptable Information Systems, pp. 95-119 New York: Springer.
33. Voorhees, EM. (1999). "Natural Language Processing and Information Retrieval," in Pazienza, MT (ed.), Information Extraction: Towards Scalable, Adaptable Systems, pp. 32-48, New York: Springer.
34. Whinston B. A., Holsapple W. C., Bonczek H. R., (1981). Foundations of Decision Support Systems , New York: Academic Press, Inc.