RC22063 (W0105-013) May 18, 2001
Computer Science

IBM Research Report

SupportBeam: An Infrastructure for Web-based Customer Support

Ajay Mohindra, Thomas E. Chefalas, Alexei A. Karve, Quentin Gouedard, Steve Mastrianni
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

SupportBeam: An Infrastructure for Web-based Customer Support

Ajay Mohindra, Tom Chefalas, Alexei Karve, Quentin Gouedard[1] and Steve Mastrianni

IBM Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598

Abstract

Telephone-based customer support has been a major source of expenditure for most companies. In recent years, with the growing popularity of the Internet and the World Wide Web, companies have started exploring ways to reduce telephone-based customer support costs by directing customers to a website. In this paper, we present the architecture, design and implementation of SupportBeam, an infrastructure to provide web-based customer support. SupportBeam uses a rule-based approach to provide a personalized web experience to users seeking answers to their questions. We use applicability rules to specify relationships between documents and products, and evaluate these rules at runtime to generate a personalized web experience. Our approach provides a scalable and flexible solution for web-based customer support with an improved user experience at low cost.

[1] Quentin was a co-op student at IBM T. J. Watson Research Center.

1. Introduction

Telephone-based customer support has been a major source of expenditure for most companies. According to industry estimates, on an average, a telephone call to the support desk costs a company between \$20 - \$30 [1]. The support-related expenses have started to affect a company's bottom line. To reduce telephone-based customer support costs, some companies have started charging customers on a per-call or a per-incident basis. In recent years, with the growing popularity of the Internet and the World Wide Web, companies have started exploring alternative ways to reduce support costs by directing customers to the company's website for support. The intent behind the support website is to provide customers with information most frequently requested on the telephone support desk. The advantage of a support website to customers is that information is accessible on a 24x7 basis. The advantage to a company is that the website reduces operational costs associated with the telephone support center. The resulting savings from migrating to a support website are substantial in spite of some investment needed to operate and maintain the infrastructure associated with the website.

The effectiveness of a support website can be determined by the degree with which it reduces the number of calls to the telephone support center. This reduction is determined by the quality and ease of use of the website, the currency of available information, scalability, and robustness of the infrastructure, and ease of authoring and publishing of content on the website. In this paper, we describe the architecture, design, and implementation of SupportBeam – an infrastructure for deploying web-based customer support. We have deployed the infrastructure for handling support requests for PC hardware and software. The infrastructure can be easily extended to provide support for other types of products. The rest of the paper is organized as follows. Section 2 describes the vision and architecture of SupportBeam, Section 3 describes the underlying information model, Section 4 discusses the implementation and performance, and Section 5 presents the conclusions and future work.

2. Vision and Architecture

The vision of SupportBeam is to provide a complete end-to-end infrastructure for reducing both the customer total cost of ownership and the support costs while delivering the best service possible to customers. The basic premise upon which SupportBeam is built is -- *If a business knows what it offers then it can effectively fulfill its customer's needs by knowing who the customer is and what the customer has*. Figure 1 shows the architecture of SupportBeam. It consists of four logical parts described below.

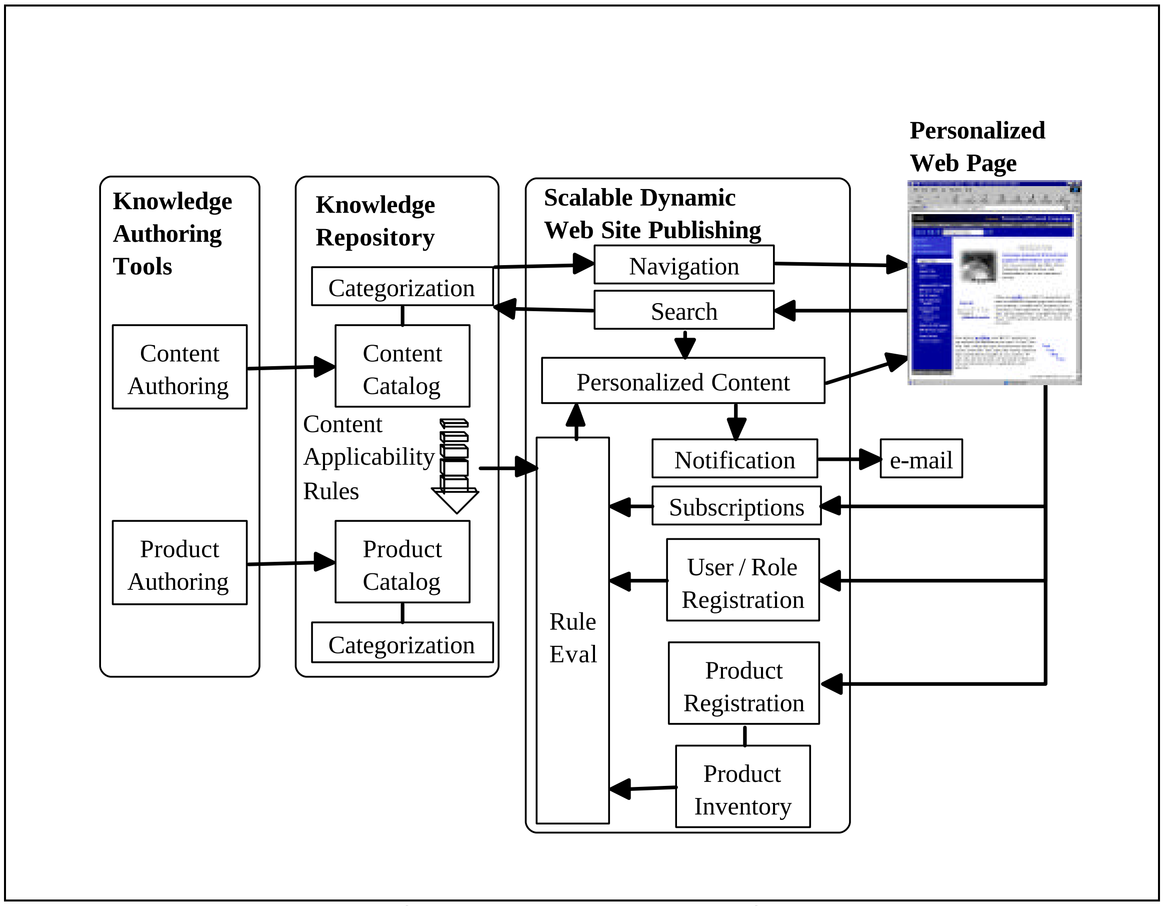

Figure 1. The figure shows the architecture of SupportBeam

Personalized web page

Providing a personalized web experience is the main "mantra" in SupportBeam. When a user visits the support website, she has the option of either browsing the website as an anonymous user or logging onto the website. When logged in, the user receives a personalized website experience based upon information available in her user profile. The user profile consists of the contact information and a list of products along with installed applications the user owns or manages. For example, a user could register that she owns a *ThinkPad T22* with *Microsoft Windows 2000* operating system and *Microsoft Office 2000* suite installed. The user profile also contains subscriptions to newsletters and discussion newsgroups, and an option to be notified whenever information matching the profile is published on the website.

Website infrastructure

The website infrastructure consists of several components. The *User/Role Registration* component enables users to register their contact information and their role in a user profile database. Roles allow a user to separate products that they own or manage in various capacities. A user selects a role when she logs in. A user logged as an **admin** sees device driver updates, while logged as an **end-user** sees only application updates. After the user profile has been created, a user can add products they own or manage to their product inventory. Information contained in the user's product inventory is used to personalize the website. For example, a user, whose inventory only contains one *ThinkPad T22* notebook, sees documents applicable only to that notebook and does not see information related to any other models.

At the heart of SupportBeam is a *Rule Evaluation Engine* that generates personalized content based upon the information available in the user profile. Each time a user visits the website, the rule evaluation engine determines the set of documents in the *Personalized Content* component for display. To simplify navigation and search, *Category based Navigatio*n and *Searc*h components are available. Each user can also subscribe to newsletters and discussion newsgroups via the *Subscriptions* component. The *Notification* component allows the users to be notified whenever information matching their profiles is available on the website.

Knowledge Repository

The knowledge repository contains both the Content catalog and the Product catalog. The Content catalog contains all the support documents available on the website. Examples of such documents in the Content catalog include documents describing bug fixes, frequently asked questions, troubleshooting guides, tutorials, and product descriptions. The Content catalog also contains software such as device drivers, and BIOS updates published on the website.

The Product catalog contains the description and specification on all the products that the website provides information about. In SupportBeam, the products include different models of PCs along with their constituent parts such as CDROM drives, hard disks, keyboards, mice, monitors, cables etc. It also includes all software packages including operating system, applications, and device drivers. The data in the Content catalog is linked via *applicability rules* to other items such as products, roles, discussion newsgroups, and categories. Documents in the Content catalog are related to other items via *applicability rules.* For example, a document describing a device driver update for *Matrox* display adapters is linked via an *applicability rule* to PCs that contain a *Matrox* display adapter, a category named “Display Adapters,” a discussion newsgroup on display adapters, and the role of a *Service Technician*. The *applicability rules* are specified when a new document is authored in the Content catalog.

Knowledge Authoring Tools

The effectiveness of personalization provided by the web site is as good as the metadata available in the knowledge repository. SupportBeam provides tools to enable authors to easily create and publish information in the Content and the Product catalogs. The tools utilize an information model described in Section 3. The tools require the author to specify the *applicability rules* for each document in the Content catalog, and the category for each document in the Content and Product catalog. Categorization is used to simplify navigation and presentation of information to users of the support website.

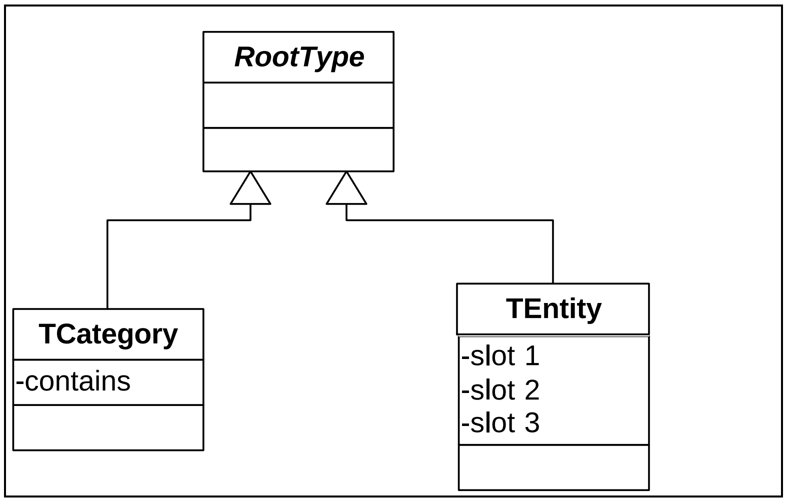

Figure 2 Class diagram for the Information Model

3. Information Model

The main challenge in SupportBeam was to design an information model that was simple, yet flexible and extensible to handle diverse data sets. The primary goal of the information model was to allow for fast and accurate retrieval of information, and to simplify creation and evolution of mass content. The structure of the information model was intentionally kept separate from the presentation logic to allow dynamic publication of content on the website.

In SupportBeam, we use a frame-based representation [2] for information stored in the knowledge repository. A *frame* provides a structured representation of an object or a class of objects. A *frame* can include sets of attribute descriptions called *slots*. A *slot* describes attributes of the object or class represented by the frame.

Figure 2 shows the class diagram for the information model. In SupportBeam, an abstract class called **RootType** represents the root *frame* in the information model. Both the abstract and the concrete *frame* objects inherit from the **RootType** class. This class cannot be instantiated and provides methods that are common to other objects. The **RootType** class has a property that each instance created from the root class or any of the derived classes is assigned a globally unique identifier. The identifier is used for bookkeeping purposes to keep track of instances at runtime.

The other two *frame* classes, **TEntity** and **TCategory**, are concrete classes that inherit from the **RootType** class. The **TEntity** class represents concrete object types. It is sub classed to create application specific classes. An instance of a **TEntity** class is called an *entity*. A *slot* in the **TEntity** class and its derived classes defines a property associated with the class. A *slot* contains the *type* of the property, its *name*, and its *value.* Classes derived from the **TEntity** class inherit the slots defined in the parent classes.

Figure 3 shows an example of a **TEntity** class. In Figure 3, a *frame* class *CD-ROM*, representing cd-roms, inherits from the **TEntity** class, and contains two *slots* -- one called *Speed*, representing the speed of the cd-rom drive, and the second called *Manufacturer*, representing the manufacturer of the cd-rom drive. The values for these *slots* are assigned when instances of the CD-ROM *frame* class are created. The figure shows two instances named “Creative” and “Teac.”

Figure 4 shows a more complex class hierarchy. In the figure, the **TEntity** class is the parent for classes **Hardware** and **Software**. Class **Hardware** is further specialized into three classes: **System, Storage,** and **Components.** The **Storage** class is further specialized into three classes: **ROM, Hard Disk,** and **Floppy Disk.** Class **ROM** is further specialized into two classes: **CD-ROM,** and **DVD-ROM**. Any new *slots* introduced during specialization are automatically inherited by the derived classes. The reader can see that the class hierarchy can be arbitrarily complex.

The second *frame* class used in SupportBeam is called **TCategory.** The **TCategory** class defines the type of abstract concepts an entity may be associated with. The **TCategory** class contains *slots* that represent the **contains** relationship. A class derived from the **TCategory** class is called a *category*. A *category* represents a powerful way of filing and retrieving precise information. A *category* name is not unique as it can be contained in other categories. Figure 5 shows an example of four *category* types: *Brand*, *Family*, *Machine Type*, and *Model Number*. For each category, the **contains** slot has been assigned a value represented by an arrow. For example, the category of type *Brand* can only contain category of type *Family*. Category of type *Family* can only contain a category of type *Machine Type*, and a category of type *Machine Type* can only contain a category of type *Model Number*. In the figure, category *IBM ThinkPad* can only contain categories *600* and *700*. Categories *600* and *700* can only contain categories *2645*, *2647*, *2800*, and *2801*, and so on.

4. SupportBeam Implementation and Performance

The SupportBeam infrastructure is implemented using Java on the IBM WebSphere Application Server 3.5 [3] and DB2 7.2 [4] platforms. Figure 6 shows the multi-tiered architecture of the implementation. All the enterprise data, including the Content catalog, the Product catalog, and user profiles, is stored in DB2. The user registration information is stored in a centralized LDAP [5] based directory. The LDAP repository also performs user authentication. The server-side business logic is implemented using session and container Enterprise Java Beans [6] deployed on the WebSphere Application Server. The server-side presentation layer uses the Java Server Pages and Servlets to construct HTML pages that are sent to the clients. A user uses a standard web browser to view and browse the support website. Java based tools are available for authoring and publishing information to the Content catalog and the Product catalog. These tools use the Enterprise Java Beans to manage information in the enterprise data repositories. By separating the structure of the information model from the presentation logic, SupportBeam enables authors to dynamically publish new content to the website without changing the presentation logic.

4.1 The rule engine

The heart of SupportBeam is a rule-based engine that performs personalization of content for each user. Without personalization, the SupportBeam infrastructure is no different from a standard website where the user is responsible for navigating and searching through thousands of documents to find relevant information. As mentioned earlier, each document published in the Content catalog has associated with it *applicability rules* that specify relationships between the document and other items, such as products, roles, discussion newsgroups and categories stored in the database.

When a user visits the support website, she has the option to anonymously browse the website or logon to the website. When anonymous, the user can browse and search all the content on the website. No website personalization is available when a user is anonymous. A user also has an option to register and create a user profile with the website.

If a user logs on, the website authenticates the user from the directory server and retrieves the associated user profile. The user profile contains list of products that the user owns or manages along with any other interests specified by the user. Using the product specification, the rule engine evaluates the *applicability rules* associated with the documents in the Content catalog. The results of the rule evaluation determine the documents that are applicable to the user. These documents are organized into categories, and presented to the user.

4.2 Inference of applicability for product-category association to documents

An author creates associations between a document and a set of categories. The associations allow the rule engine to infer the applicability between documents and product/categories through upward and downward propagation. This applicability of documents to multiple categories is saved in a cache called the Bucket Cache. Figure 7 shows an example of inference of applicability for product-category association to documents. The rectangular boxes represent different categories created by instantiating classes: *Brand, Family, Machine Type, and Model Number.* Numbers from 1-13 represent documents in the Content catalog. Numbers inside the rectangular box represent an association authored between the document represented by the number and the category represented by the box. Numbers inside the ovals attached to the rectangular boxes represent the applicable documents that are saved in the Bucket Cache.

In the example, an association is authored between documents 6 and 7 to category *Machine Type* "6889". It means that documents 6 and 7 are applicable not only to category *Machine Type* "6889" but also to any of the ancestor categories of *Machine Type* "6889" and to any of the descendents (represented by the **contains** relationship) of *Machine Type* "6889". The ancestors of *Machine Type* "6889" are *Family* "IntelliStation MPro" and *Brand* "IBM IntelliStation." Therefore, the Bucket Caches of both these ancestors will contain the documents 6 and 7. Since *Family* "IntelliStation Zpro" and *Family* "Intellistation Epro" are not ancestors of *Machine Type* "6889", documents 6 and 7 are not propagated to them. Documents 6 and 7 are applicable to all the descendents of

Machine Type "6889" which are *Model Number* "15K", "15U" and "15V" and all the associated instances of the products. In this example, only multiple descendents are shown. However, a category can have multiple ancestors (can be contained in multiple other categories). Therefore, the propagation will take place upwards through all ancestors.

A product, in turn, is a complex entity and is made up of other entities, for example, display adapters, hard drives, monitors, and processors. If a document is associated with a particular display adapter and that display adapter is a part of another product then the document is propagated upwards to all ancestors of the product containing the display adapter.

	Number of items
Categories	41,000
Products	28,000
Entities	2,900,000
User profiles	863,000
Applicability rules	3,200,000

Table 1. Data distribution in SupportBeam for the performance benchmark

	Time
Cold start	20 seconds
Warm start	< 1 second

Table 2. The table shows the response time to display a personalized web page to a user after she has selected a product from the inventory.

4.3 Performance Measurements

We have performed some preliminary benchmarking to gauge the performance of SupportBeam. The performance was measured on the following configuration. The WebSphere Application Server is running on a dual-processor *Netfinity 5600* machine with 1.3 GB memory. The enterprise data server is a DB2 server running on a four-processor *Netfinity 56*00 machine with 2 GB memory. The total size of support data is 1.87 GB. Table 1 summarizes the distribution of data in the database. The infrastructure contains 28,000 products, classified into 41,000 categories. There are 2.9 million entities in the database, which includes all products, support documents, software updates etc. These entities are linked by 3.2 million applicability rules. The total number of user profiles stored in the system is 863,000.

We measured the response time for a user to be presented with a personalized web page. This operation involves the following steps.

1. A user logs on the website by entering username and password
2. The runtime authenticates the user.

3. After authentication, the runtime retrieves the user profile and associated product inventory.
4. The contents of the product inventory are displayed to the user.
5. After the user selects a product from the displayed list, the runtime computes the applicability rules for that product, retrieves relevant documents, organizes the documents into available categories, and presents the results to the user.

We measured the response time to complete Step 5 above. Table 2 shows the results of the performance measurement. The measurement shows the response time to display a personalized web page to a user after she has selected a product from the inventory. For a cold start, the response time is about 20 seconds. This response time is large because the WebSphere Application Server has to load all the Servlets and Enterpsie Java Beans, and the DB2 subsystem has to be primed. After the website is operation, in a warm start, the response time is reduced to less than 1 second.

5. Conclusions and Future Work

In this paper, we present the architecture, design and implementation of SupportBeam, an infrastructure to provide web-based customer support. SupportBeam uses a rule-based approach to provide a personalized web experience to users seeking answers to their questions. We use applicability rules to specify relationships between documents and products. We separated the structure of the information model from the presentation logic to enable authors to dynamically publish content to the website without changing the presentation logic. Our initial performance results are promising and show that better user experience can be provided via personalization at a reasonable cost.

In the future, we plan to perform a detailed performance tuning of the system, and optimize the evaluation of applicability rules, and the Bucket Cache. We also plan to perform system load tests to measure the response time when multiple concurrent queries are performed.

References

1. eWeek. “Service bots are hot.” In eWeek Magazine, April 16, 2001. http://www.zdnet.com/eweek/stories/general/0,11011,2706274,00.html
2. Minsky, Marvin. “*A framework for representing knowledge*”. In: P. Winston (ed.) The Psychology of Computer Vision. New York: McGraw Hill: pages 211-7.1975
3. IBM WebSphere Application Server *http://www.software.ibm.com/*
4. IBM DB2 http://www.software.ibm.com/
5. Lightweight Directory Access Protocol. RFC2251. December 1997.
6. Enterprise Java Beans http://java.sun.com/products/ejb/

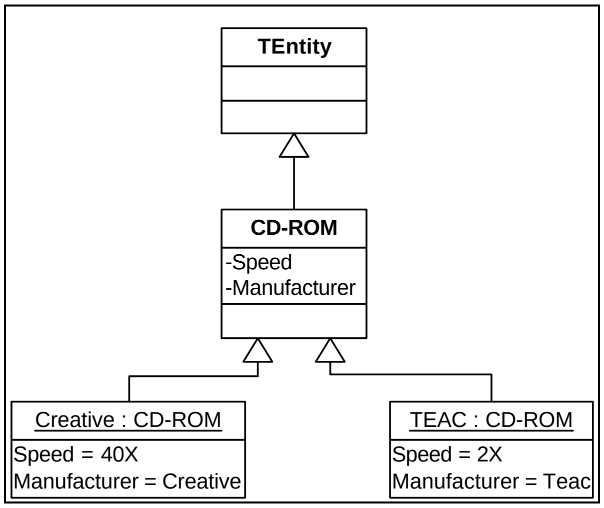

Figure 3. An example of TEntity and entity

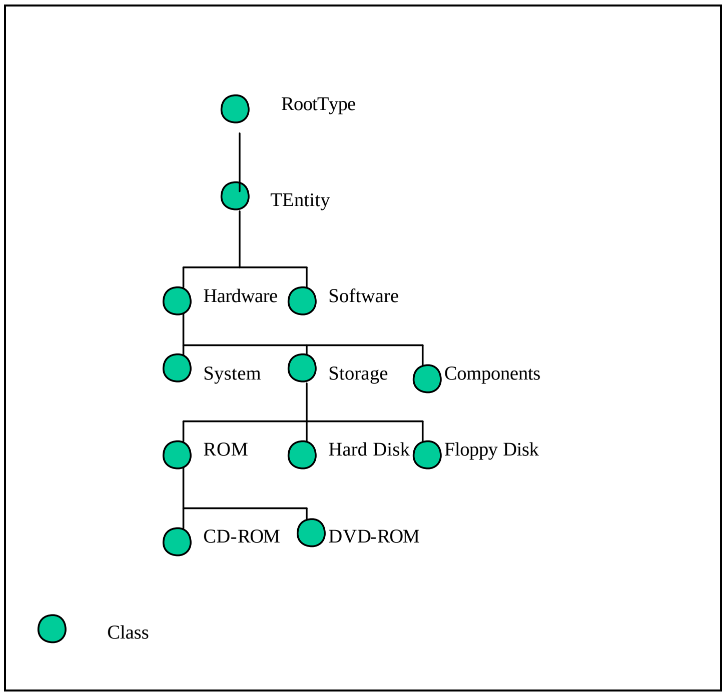

Figure 4. An example of a TEntity class hierarchy

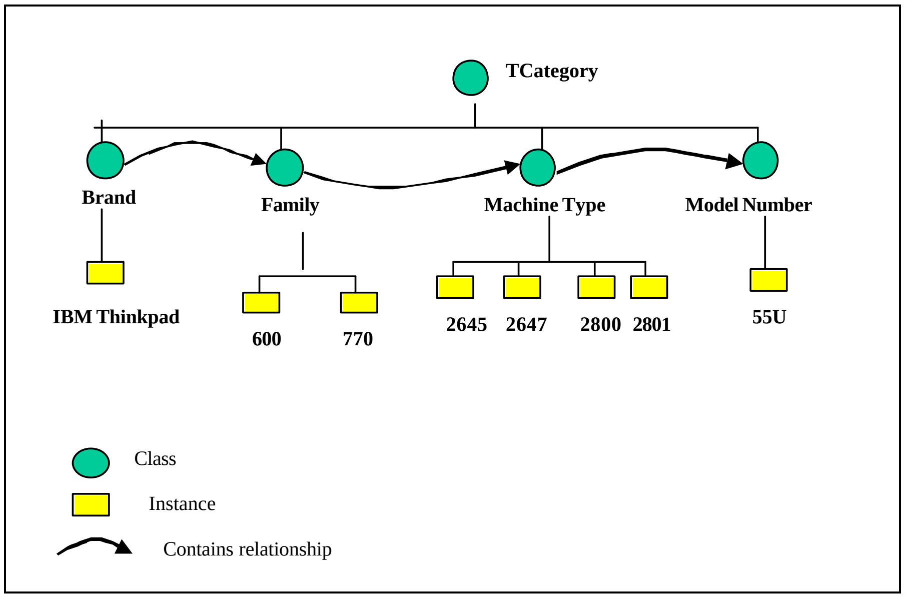

Figure 5. Examples of TCategory and Category

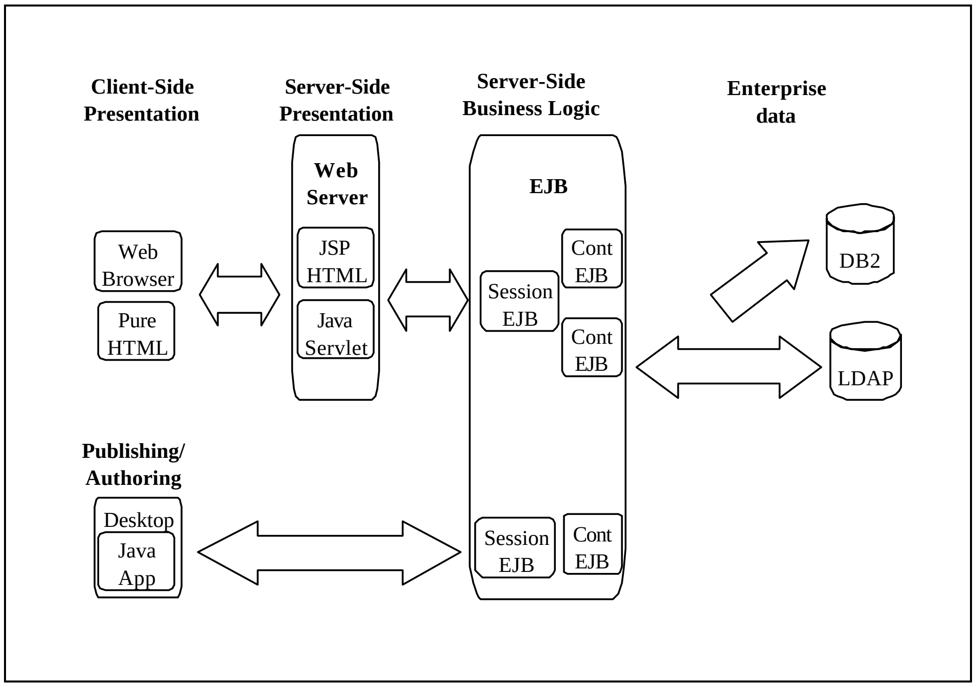

Figure 6. Multi-tiered logical diagram for SupportBeam implementation

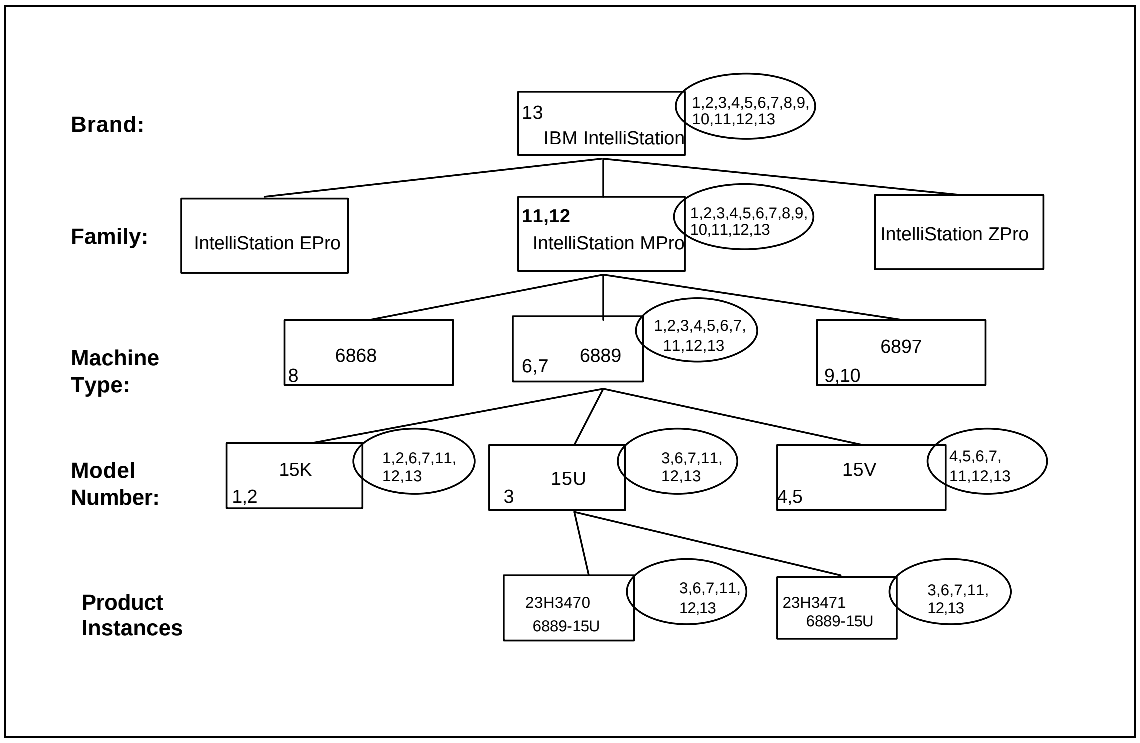

Figure 7. The figure shows an example of inference of product-category associations for documents. The rectangular boxes represent different categories created by instantiating classes: *Brand, Family, Machine Type, and Model Number.* Numbers from 1-13 represent documents in the Content catalog. Numbers inside the rectangular box represent an association authored between the document represented by the number and the category represented by the box. Numbers inside the ovals attached to the rectangular boxes represent the applicable documents.