

# NetPal: A Dynamic Network Administration Knowledge Base

Ashley George, Adetokunbo Makanju,  
Evangelos Milios, Nur Zincir-Heywood  
Faculty of Computer Science  
Dalhousie University  
Halifax, NS  
{ageorge, makanju, eem, zincir}@cs.dal.ca

Markus Latzel, Sotirios Stergiopoulos  
Palomino System Innovations Inc.  
Toronto, ON  
{markus, sotirios}@palominosys.com

## Abstract

Netpal is a web-based dynamic knowledge base system designed to assist network administrators in their troubleshooting tasks, in recalling and storing experience, and in identifying new failure cases and their symptoms. In the context of web hosting environments, Netpal summarises network data and supports retrieval of relevant organisational experience for system administrators. The system design draws on a variety of domains including knowledge management, information retrieval, machine learning and network management. We describe the system architecture, user interface design, user software testing and future directions for development.

## 1 Introduction

In the course of their duties, system administrators acquire and retain extensive specialised knowledge regarding the operation of complex systems and networks. Experienced system administrators can rapidly narrow the field of candidate solutions when confronted with network faults and associated evidence. Thus the value of experience for a system administrator is high; the administrator uses this knowledge

to judge an appropriate initial avenue of investigation. This presents an opportunity to optimise use of time and resources by creating a system which supports experience management [2, 8, 9] and recall in system administrators through suggestion of relevant previously recorded experience cases.

In addition to knowing how to solve a recognised fault, an administrator needs to detect the evidence which characterises a fault. The collective data stream generated by hosts in a large network grows to exceed that which can be sorted by ad-hoc visual inspection. Numerical and visual tools for filtering and reducing the data stream allow the administrator to gain a high-level overview of their network's behaviour.

In the NetPal project, a prototype experience retrieval system for system administrators was produced in the context of the web hosting business. Our goal is to reduce the learning curve faced by junior system administrators and to supplement recall in senior system administrators through retrieval of previous experience cases, integrated with web-based reporting and analysis of network event data, particularly host-based server log data. We approach this problem through a combination of information retrieval, knowledge management, host and network analysis and user interface design.

The process of troubleshooting a network fault is described in [10] as containing the following steps and roughly illustrated in Figure 1.

---

Copyright © 2008 Dr. Evangelos Milios, Dr. Nur Zincir-Heywood, Ashley George, Tokunbo Makanju, Markus Latzel, Sotirios Stergiopoulos. Permission to copy is hereby granted provided the original copyright notice is reproduced in copies made.

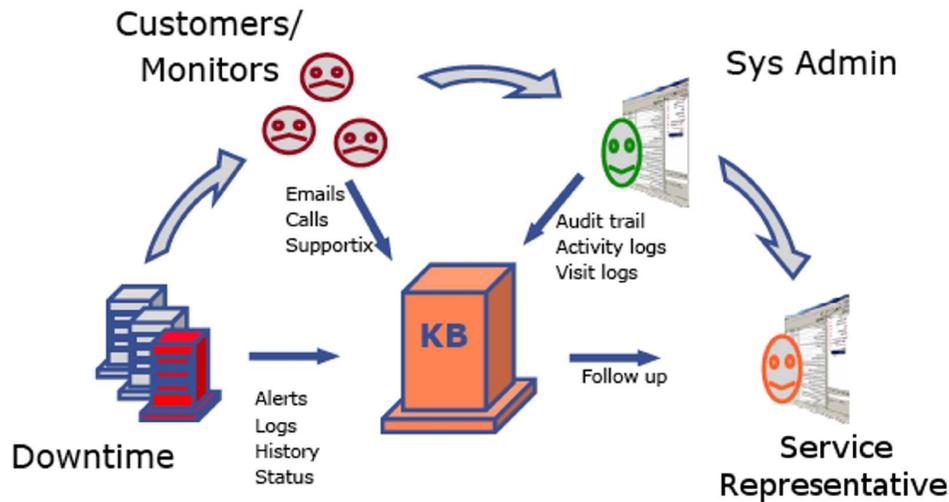


Figure 1: NetPal: network fault lifecycle diagram

1. alarm collection
2. customer satisfaction maintenance
3. alarm filtering and correlation
4. fault diagnosis
5. development and implementation of an action plan
6. verification of fault elimination
7. statistical analysis of fault management process

The fourth and fifth steps are considered to account for 80% of the troubleshooting efforts, based on experience in Palomino; reductions in the effort required from system administrators both for identifying faults and planning resolutions would result in significant savings for an organisation. We aim to achieve these reductions by increasing the amount of relevant prior information available to system administrators as they tackle a particular fault scenario.

## 2 Motivation

The NetPal project is inspired by previous efforts in constructing network knowledge management systems [4, 7, 10, 6]. Many of these

systems incorporate rule-based or case-based reasoning. We also take note of a successful experimental medical diagnosis support system, based on similar principles of retrieving literature relevant to a case using text mining techniques [12]. Drawing on these sources, we focus on constructing a system which acts to ease the recall of experience and the interpretation of system information by administrators. Therefore we are less concerned with automatically producing corrections for problems; we are more interested in prompting both learning and recall in employees who are well-suited for correcting emerging problems.

### 2.1 Rule- and Case-based Reasoning

Rule-based reasoning (RBR) systems, in the context of computer network management, tend to take a control systems approach, as reviewed in [4]. In a network-oriented control system, there is a database of rules which consists of conditions and procedures. The conditions represent knowledge about the normal operating state of the network. When these conditions are violated, the network enters an irregular operating state. At this point, procedures can be executed to ostensibly return the state of the network to the normal operating

state. In [7], Lewis suggests that rule-based reasoning systems tend to suffer from brittleness and a “knowledge acquisition bottleneck”. These faults are respectively characterised by difficulty in adapting to novel problem situations and an unconditioned growth of the rule-base as new conditions and procedures are inserted. In the latter case, the database becomes unmanageable and may lead to unpredictable behaviour.

Following this assessment, Lewis proposes a case-based reasoning system to record and adapt “episodes of problem solving” where the adaptation of these episodes consists of observing interactions with the case database (“abstraction/respecialization adaptation”) or modifications directly to the cases themselves (“critic-based adaptation”) [7]. The system, ‘Critic’, enhances a typical trouble ticket database system by introducing constraint-based ticket filtering and user feedback-capturing mechanisms for updating the case database over time while minimising user intervention where possible. Lewis’ augmentation of the trouble ticket system appears to favour a predicate-oriented representation (as with RBR) where we prefer an information retrieval strategy.

An alternate approach to case-based reasoning for system administration is to insert the reasoning system at a different point in the problem resolution process. In [10], the case-based reasoning system is situated closer to the alarm collection source, in the sense that alarms are processed by the system before a ticket is raised, in the hopes of reducing the volume of raised tickets. This approach targets a coarse-grained control system - when the collected alarms match the preconditions for a case, that case and its attached procedures are automatically applied to resolve the problem. Contrast this coarse-grained activity with the application of a discrete series of individual rules and procedures geared to incrementally restore operational state, as in a rule-based system. For their experimental testing, pre-defined cases were used. It was found that procedures were automatically applied to 8% of faults over time, leaving 92% of faults for “in-site maintenance”.

Each of the systems described in these works

focuses on applying fine-grained or coarse-grained procedures in the context of telecommunications systems when predetermined conditions are matched. In the fine-grained case, the expert is required to craft a number of condition-action statement pairs which incrementally return the system to its ideal operating state. In the coarse-grained case, the domain expert crafts a complex condition and a complex procedural response which returns the system to its ideal operating state. In both cases, the requirement of formal statements for conditions and solutions suggests difficulty in future adaptation of the rule or case base to changing network conditions.

## 2.2 Information Retrieval and Medical Diagnosis Support

Patients in a clinical setting may exhibit a mixture of symptoms, giving rise to multiple plausible explanations for their condition. The caregiver for the patient is expected to infer from the observed symptoms (and patient case history) an accurate diagnosis which will provide a basis for corrective and preventative medical action.

In the context of medical diagnosis, ISABEL [12] is a diagnosis support system which, in early trials, has reduced errors of “diagnosis omission” [11, 16, 13]. ISABEL is reminiscent of a search engine and is heavily tailored toward searching medical literature. This content restriction allows for the implementation of domain-specific features, including presentation of probabilistically ranked diagnoses, expert annotations of case material and more. Unlike the rule-based and case-based systems described in the previous section, the purpose of ISABEL is not to automatically apply rules or case-based scripts but to act as a reminder aide for the expert.

Much as a doctor may benefit from prompting to consider alternative diagnoses, network administrators may also benefit from a system which prompts knowledge recall. Our problem domain requires a different approach from ISABEL given that there is no systemic, widely applicable body of diagnostic literature which uniquely characterises root causes of fault symptoms for web hosting environments.

Thus we provide the administrator with search and visualisation tools to help them interactively narrow their hypotheses.

We aim to enhance the abilities of administrators to filter generated system events and to recover previous work experience. Where rule-based and case-based systems attempted to automatically detect conditions and apply scripted solutions, we rely on arming administrators with increased prior information so they may better assess conditions and produce solutions. Where ISABEL links medical literature to symptom descriptions, we index and present previous experience and domain-specific documents to aid the administrator in gaining understanding of the current case. We aim not to automate all problem-solving but to improve knowledge transmission and recall for junior and senior network administrators.

### 3 The Web Hosting Environment

In our work, NetPal is developed and tested in the context of the small to medium web hosting environment. Servers deployed in a web hosting environment handle hosting responsibilities in a distributed manner. In addition to web content, there are many services offered by the web hosting provider to support their customers, including email, domain name services and custom application support. The web hosting environment therefore produces multiple kinds of faults, emanating from different hardware, software, customer and consumer sources.

The hardware and software configuration selected by a web hosting provider will depend on customer needs. Cardellini et al [3] describe several configuration scenarios for handling high-load web applications. While small customers with relatively static website content consume a fraction of a server's capacity, customers with high traffic and application processing costs require concurrent solutions consuming multiple physical servers, both for processing and for redundancy.

Allocation of resources in a web hosting environment is therefore multiplied over reliability and responsibilities. To handle these respon-

sibilities, there are typically groups of servers dedicated to

- handling web requests
- application layer processing
- database and object retrieval

NetPal targets this kind of heterogeneous operating environment which contains off-the-shelf components, custom components and per-customer hardware and software configurations. By taking an information retrieval approach, we hope to mitigate some of this heterogeneity when searching through collected data. For our testbed environments, we chose to simulate a complex hosting environment by replaying log data and case history collected at Palomino Inc. into databases on the NetPal test servers. In Section 5, we further describe how our software testers interacted with this system and summarise their feedback.

### 4 System Architecture

The NetPal system is composed of a set of core components which are intended to integrate with external fault detection systems. The core components include the following.

- case knowledge base
- problem/solution matching engine
- presentation, feedback and editor modules

Their component interactions are summarised in Figure 2.

#### 4.1 Case and Problem/Solution Matching

Underlying the case knowledge and problem/solution matching engine is an information retrieval (IR) subsystem. In an information retrieval system, the prevailing metaphor is that of query-answering, normally by returning a ranked set of documents for the user's query under a measure of relevancy [1, 5].

In our system, queries can be generated manually or automatically as the system administrator interacts with NetPal and builds a case.

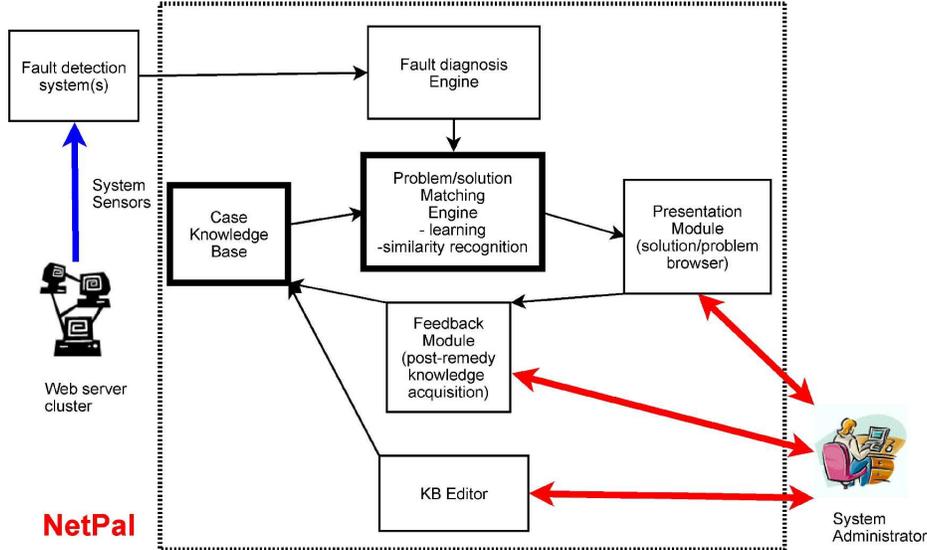


Figure 2: NetPal core system components diagram

We then make recommendations for related articles, cases and events (logfile entries, at this time). The administrator can also query the relevant components individually. We would typically generate a list of keywords from case components but an example of a manual sample query might be a series of keywords garnered from observing log entries or a best initial hypothesis, such as

- “php open\_basedir”
- “domain conf problem”

These kind of queries can be extended through query expansion: using a thesaurus with domain knowledge to automatically produce a list of synonyms. We use query expansion to optionally broaden the scope of queries generated in NetPal.

#### 4.1.1 Information Retrieval

We adopt the vector space model to characterise our cases for retrieval purposes and to capture the similarity between vectors using the cosine angle distance. This model, originally described in [14] and summarised in [5], represents documents as vectors in a term frequency-inverse document frequency (TF-IDF) space. We describe this model here as a basis for our system.

Let  $T$  name the vocabulary of unique terms extracted from the document corpus and  $t_i$  be the  $i$ 'th term in  $T$ . Let  $D$  be the set of all documents and  $d_j$  be a particular document. Then, the normalised term frequency  $tf_{ij}$  is given by

$$tf_{ij} = \frac{n_{ij}}{|d_j|} \quad (1)$$

where the frequency of occurrence of term  $t_i$  in document  $d_j$  is denoted by  $n_{ij}$ . This measure captures the relative importance of a term in a particular document while accounting for the effect of varying document length.

The in-document term frequency (TF) is one component of the representation. The other is the inverse document frequency (IDF). Given a term  $t_i$ , the inverse document frequency is calculated as

$$idf_i = \log \left( \frac{|D|}{|\{d_j : t_i \in d_j\}|} \right) \quad (2)$$

This quantity characterises the support of the term  $t_i$  in the document corpus; terms which appear in few documents obtain a high IDF score while terms which occur widely throughout the document set will score closer to zero.

For document  $d_j$  and term  $t_i$ ,  $tfidf_{ij}$  is the product of these quantities. The term frequency captures the in-document importance

for  $t_i$  and the inverse document frequency captures the whole-corpus importance for  $t_i$ . These values, multiplied, reward most those terms which are highly characteristic of a particular document, yet not widely occurring with respect to the entire document set. Finally, a particular document  $d_j$  is represented as a vector of such TF-IDF scores, with one entry for each term in the vocabulary  $T$ , as in Equation 4.

$$tfidf_{ij} = tf_{ij} \cdot idf_i \quad (3)$$

$$d_j = \{tfidf_{(1,j)}, tfidf_{(2,j)}, \dots, tfidf_{(|T|,j)}\} \quad (4)$$

We facilitate retrieval of document vectors by constructing an inverted index [1, 5] relating terms to the documents in which they appear. This enables a query document to be matched against terms in the database which in turn generates a set of candidate matching documents. That is, for each term in the query document, the corresponding set of documents in which that term appears is fetched from the database. However, these document sets may overlap and they are unordered. Therefore, we order them based on the cosine angle similarity between the query vector,  $q$ , and the vector for each document in the result set.

$$cossim(q, d_j) = \frac{\sum_i tfidf_{iq} \cdot tfidf_{ij}}{\sqrt{\sum_i tfidf_{iq}^2} \cdot \sqrt{\sum_i tfidf_{ij}^2}} \quad (5)$$

The query vector can be generated manually or automatically from different sources; we adapt the information retrieval system to different tasks through modifying the indexing and either requesting query input or generating queries automatically.

#### 4.1.2 Documents and Queries in NetPal

We described the vector space model for information retrieval. In NetPal, we use this model for retrieving indexed articles, cases and events. Each of these type of documents is represented as a vector of *tfidf* weight values in a corresponding index. We reduce the index size by filtering unrepresentative terms, as is common

in information retrieval applications [1, 5]; in particular, we filter terms by their inverse document frequency, reducing the term dimensionality of our indices by an order of magnitude. (e.g., from tens of thousands of terms to a thousand terms.)

We produced a system taxonomy through a combination of automatic generation and expert refinement of the vocabulary, and with manual organisation of the taxonomy’s hierarchical structure. We use the taxonomy to expand queries by introducing related terms. For problem/solution case matching, we generate queries automatically from terms in the case itself and any attachments it may hold (including text fields, log events and attached articles or cases). When directly searching through cases, we expressly allow the user to input queries and optionally expand these using the taxonomy.

#### 4.1.3 Log Data Acquisition Engine

We developed a system for aggregating logged data, dubbed the ‘Log Data Acquisition Engine’ (or LDAE). The LDAE is a background process which serves to collect system log data from multiple hosts, process the entries to allow basic grouping of like events, and feed the data into an SQL database for manual or automatic correlation with cases in NetPal.

Feeding log entries through an aggregator enables the collection of global log data statistics. We initially attempted using an association rule mining algorithm for identifying strong subpatterns in log entries, as in Vaarandi [17, 15]. This would allow us to perform grouping of log entries in the user interface by frequent shared tokens. However, the offline cost of this algorithm was prohibitive for our system.

Therefore, we selected a static set of regular expressions for matching common logfile patterns. This included dates, times, IP addresses, common services, local hostnames and more. By replacing highly-variant substrings in consecutive entries with common tokens, we could enable a limited version of the grouping effect we aimed to achieve. We employ this in the user interface for ‘collapsing’ like entries into clusters and reducing the number of log entries on display.

In sum, the role of the LDAE is to collect and preprocess log data, from multiple services distributed across hosting servers, and to aggregate this data in a central database. Once these log entries are collected in the central database, they can be indexed for subsequent retrieval in NetPal.

## 4.2 User and Web Interface Design

The main NetPal interface consists of a web-based module which can be integrated in popular web hosting control panel software as a plugin or can be run separately as a standalone web application. Figure 3 displays an overview of the NetPal user interface. A number of components are visible and we address these here.

The main page is divided into two panes. We focus on the left pane which contains the event browser and event graph. The event browser is pictured in greater detail in Figure 4 (a). The browser includes several features to aid in sifting through logged data. Like cases, events are searchable and the list of relevant events can be narrowed by entering search terms in the search field at the top of the event browser. Each event is initially depicted using a one-line summary, here containing the date/time of occurrence, the service responsible for generating the event and the message associated with this event. Notice here that the raw event can be recalled by selecting the desired event in the browsing list; the event then is displayed in its entirety below the list.

Underneath the events browser is the events graph, depicted in Figure 4 (b). The events graph is meant to impart the level of event activity over time. The height of the peaks on the chart corresponds to the number of events occurring during that time interval. The time interval of the graph can be modified for greater detail. The graph is color coded, as in the legend, to indicate which services generate which peaks. Note that clicking anywhere within the graph repositions the event browser's listing to show events surrounding the chosen time.

The left-hand pane also has multiple tabs at the top. The default pane contains the event browser and event graph. Alternate panes include search interfaces for cases and articles,

a visual representation of the domain taxonomy (a text-based tree component) and the "vital signs" component. The vitals component indicates the proportion of errors recorded in each logfile on the main host and commonly requested system statistics. The vitals component is depicted in Figure 5.

When a new case is opened or a case is retrieved from the case base, the case editor/viewer, as in Figure 6, is populated with data in each relevant field either by the database or by the user. The user can populate a case by interacting with the event search in the left-hand pane, including dragging relevant events from the left pane to the right or adding them at the click of a button. As fields are filled, events added and cases linked to the current case, a heterogeneous data structure representing the case is constructed internally. This structure is stored as the user modifies it and is indexed for matching and retrieval.

## 5 Software Testing

As our prototype neared completion, we sought software testing feedback from candidate users about the usability and suitability of the software for two real-world system administration tasks reconstructed from experience by staff at Palomino Inc. At Dalhousie University, we obtained feedback on these scenarios from five testers, of which three were permanent IT staff members and two were co-operative education students working as IT helpdesk assistants. At Palomino Inc., feedback was obtained from two professional system administrators outside the company who volunteered their time to test the system.

### 5.1 Testing Configuration

At each site (Dalhousie University and Palomino Inc.), we set up a server running the NetPal system. We populated the NetPal database with sample cases, knowledge base articles and log data, previously recorded by Palomino Inc. during instances of problem-solving. We provided each tester with a software manual explaining the components of the NetPal system. The testers received scenario outlines describing observed faults which the

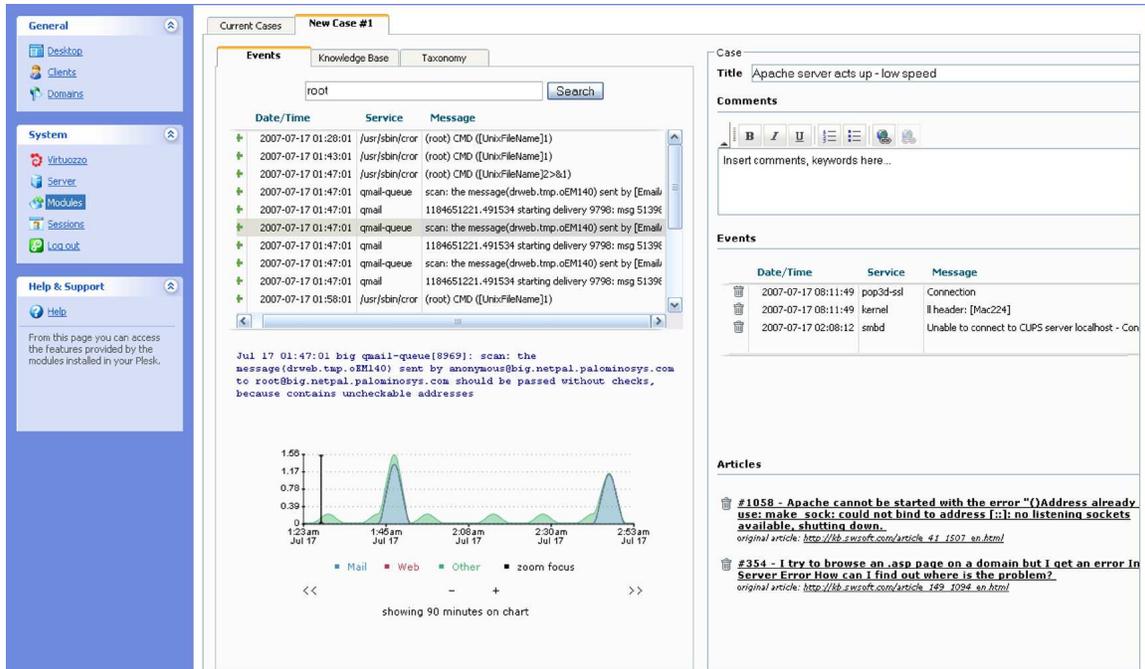


Figure 3: NetPal prototype user interface

testers would attempt to 'diagnose', given the hypothetical circumstances.

Testers were required to consider the fault symptoms, sift the log data using the web interface, build a case using NetPal, accurately identify the root cause of the observed fault and propose a reasonable solution based on experience (articles and cases) where possible. They were not required to fully implement the proposed solution. We asked the testers to describe their experiences with each scenario on a per-component basis and to provide ratings for the usefulness of each component for each scenario.

## 5.2 Tester Feedback

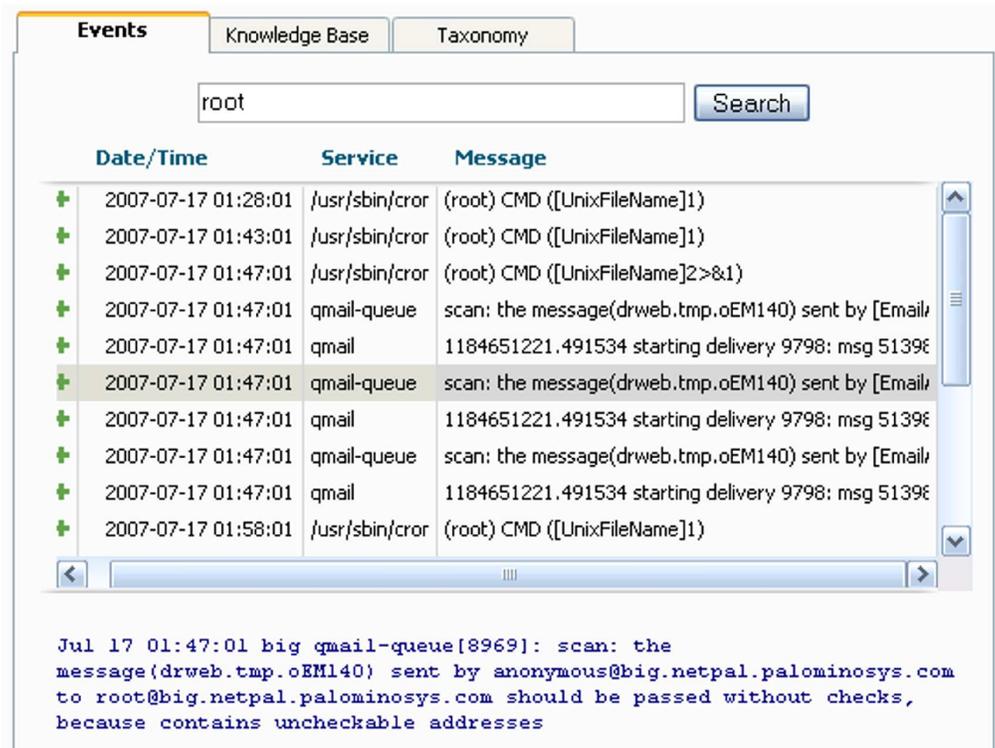
The components which we asked our testers to rate and on which we asked them to comment included the event list and graph (Figure 4, (a) and (b), respectively), the vital signs component (Figure 5), the case viewer/editor (Figure 6), the taxonomy tree and the knowledge base search.

The event list component (for searching and browsing of aggregated log data) was most

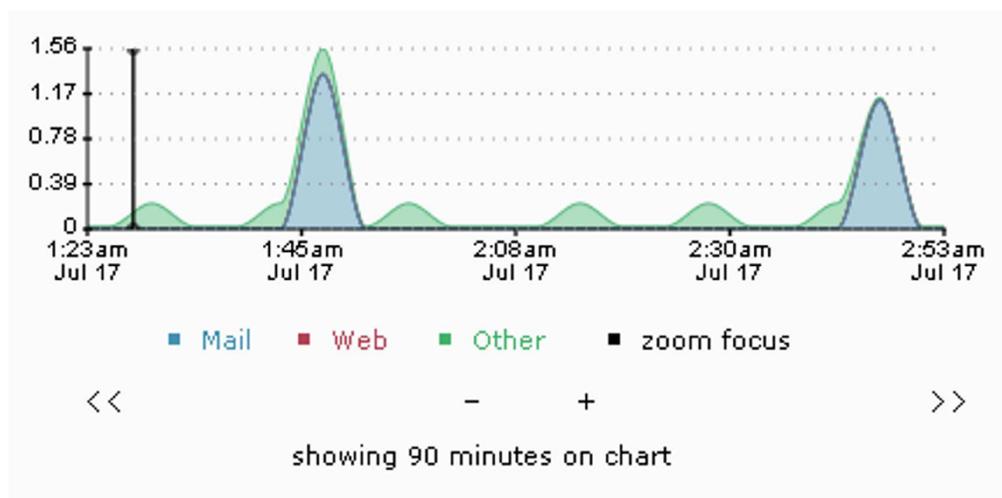
highly rated by the administrators and IT staff. The two help desk assistants gave the event listing the lowest rating. The most requested feature for the event listing was that NetPal should expose additional filtering and search features, particularly filtering on severity. The most common complaints were that there was display lag when scrolling or paging in the log event listing window and that for usability the event list should be more easily resized.

The event graph received mixed ratings. Although the majority of our testers agreed that the graph helped to situate the event listing in time and to give a sense of the period of the log data, very few indicated that they used the zooming or panning feature to support their problem solving. The reasons offered for this included the lack of resolution in the graph display and the latency of the graph refresh rate.

The vital signs component was highly rated by all testers. From the perspective of the administrative group, collecting frequently queried system data into a common display was compelling. From the perspective of the helpdesk assistants, the at-a-glance sorted list of error rate per log file was appealing. One



(a)



(b)

Figure 4: (a) The NetPal events browser and (b) the events graph

**Vitals**    Events    Similar Cases    Knowledge Base    Taxonomy

**System Info:** Linux big 2.6.16.13-4-smp #1 SMP Wed May 3 04:53:23 UTC 2006  
**Uptime:** 12:55am up 183 days 8:51, 0 users, load average: 0.29, 0.13, 0

**Disk Status:**

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/sda2	728461220	216597984	511863236	30%	/
udev	513928	116	513812	1%	/dev

**sshd**

- [/var/log/messages](#)
- [/var/log/warn](#)

**httpd**

- [/var/log/apache2/error\\_log](#)
- [/srv/www/vhosts/web1.netpal.palominosys.com/statistics/logs/access\\_log](#)
- [/usr/local/psa/admin/logs/httpd\\_access\\_log](#)
- [/usr/local/psa/admin/logs/httpd\\_error\\_log](#)
- [/var/log/apache2/access\\_log](#)

**kernel**

- [/var/log/messages](#)
- [/var/log/firewall](#)
- [/var/log/warn](#)

**imapd**

- [/usr/local/psa/var/log/maillog](#)
- [/var/log/mail.info](#)
- [/var/log/messages](#)

Figure 5: NetPal 'vital signs' component.

Case

**Title** Apache server acts up - low speed

**Comments**



Insert comments, keywords here...

**Events**

	Date/Time	Service	Message
	2007-07-17 08:11:49	pop3d-ssl	Connection
	2007-07-17 08:11:49	kernel	ll header: [Mac224]
	2007-07-17 02:08:12	smbd	Unable to connect to CUPS server localhost - Con

**Articles**

**#1058 - Apache cannot be started with the error "()Address already use: make sock: could not bind to address [::]: no listening sockets available, shutting down.**  
*original article: [http://kb.swsoft.com/article\\_41\\_1507\\_en.html](http://kb.swsoft.com/article_41_1507_en.html)*

**#354 - I try to browse an .asp page on a domain but I get an error In Server Error How can I find out where is the problem?**  
*original article: [http://kb.swsoft.com/article\\_149\\_1094\\_en.html](http://kb.swsoft.com/article_149_1094_en.html)*

Figure 6: NetPal case viewer component.

administrator requested that this component provide an expanded summary of local and network features.

Most testers suggested a middling rating for the case viewer/editor. An experienced administrator requested further automation at this point, suggesting that there be some method to jumpstart the population of a new case's fields. Another administrator requested minor interface enhancements (e.g. larger font, resizing). The rest of the testers found the viewer/editor serviceable but that the testing scenarios provided little opportunity to exercise this feature to the fullest.

We provided a display of the system taxonomy on a separate tab. This was implemented using a simple tree widget displaying the terms in the taxonomy along with their hierarchical relationships. Most of our testers couldn't relate the taxonomy to their problem-solving task. Two testers said that the taxonomy helped them to find topical synonyms in their search of the knowledge base.

Finally, we provided an interface for querying our database of collected knowledge articles and cases. Here the testers gave mixed ratings; some were able to locate relevant articles and cases and some were not. The direct search interfaces appealed to some users for the power to refine one's query directly and manually. Most requested that the search results should more strongly resemble popular search engines.

### 5.3 Discussion

The events list component and the vital signs component resonated with the administrators and IT staff. These components capture commonly made system queries. Encoding queries directly in the interface succeeds where these capture frequently used previous working experience and make it immediately accessible and replicable. So we can think of one kind of experience as accumulated patterns of system interaction which emanate from the administrator. If we can strongly capture these in the interface, we should enable the administrator to dispense with rote behaviours in their daily work. Several administrators asked for additional filtering criteria (e.g. by severity); this suggests either careful statistics or standard-

ised semantics for log files would be valuable. One tester suggested including an embedded terminal, opening an opportunity for automatically capturing command sequences for future reference.

Although the event list appealed most to senior staff, it held little interest for junior testers. It's unclear exactly how much the students were put off by network latency in the updating of the list of events. Eliminating network delays from the prototype would allow us to answer this question. It may be that a combination of inexperience and frustration prevented them from fully accessing the implicit experience represented by the events list component.

All our testers expressed some approval of the temporal and visual summary communicated by the events graph; yet, none of them found navigation via the graph to be useful in their testing scenarios. One way to improve the navigation might be to use a visualisation which directly expresses a discrete metric of the log data. The graph, as it stands in Figure 4 (b), uses a continuous function to represent frequency over time. On the other hand, events in a log file are discrete occurrences. Our testers might be better served with a discrete visualisation which can be drilled down, rather than a continuous one. (This might permit, in the extreme, the events list and the graph to be merged for additional screen real estate.)

For our test scenarios, in addition to the set of cases we crafted manually, we treated our article database as equivalent to cases for testing purposes. But, while the articles with which we bootstrapped the database are in the same problem domain (that is, web hosting), they are only an approximation of experience cases. One administrator commented that many of the articles in the knowledge base were related to initialisation or permanent configuration problems whereas the testing scenarios related more to transient problems. Additionally, the number of articles we used to bootstrap the initial database was small, on the order of a few thousand; this may have limited the applicability of the case base for our test scenarios.

## 6 Conclusion

We have described the components, development and testing of a prototype network administration information system, NetPal, which is a synthesis of information retrieval, human-computer interaction, event analysis and experience retrieval. We outlined the significant user interface components, described the inner workings of the retrieval system and our model for acquiring and processing log data for consumption in the main system. We obtained feedback from testers which will aid us in selecting future directions for the project.

Our prototype faces hurdles yet we have established a footing on which to base future development and research. Future development on NetPal will focus on automatic experience collection for the case database, improving the human factors associated with a dynamic web application, applying more sophisticated data mining techniques to our acquired event data, particularly to examine the apriori-like algorithm of Vaarandi as a clustering model and to enable enhanced visualisations, and fine-tuning our search engine for optimum retrieval.

## Acknowledgements

We acknowledge the financial support of Precarn Inc. and the Natural Sciences and Engineering Research Council of Canada.

**Ashley George** received his BCS and MCS in 2003 and 2006 respectively from Dalhousie University and is now pursuing his Ph.D. His research interests focus on machine learning, document mining, network and system security and philosophy of science.

**Adetokunbo Makanju** obtained his B.Sc. from the Department of Computer Science at University of Lagos, Nigeria and his M.CS. in 2008 from Dalhousie University. He is presently studying for his Ph.D. with the Faculty of Computer Science at Dalhousie University. His research interests are in the areas of wireless networks, intrusion detection, genetic programming and case-based reasoning.

**Evangelos Milios** is Professor and Killam Chair of Computer Science with the Faculty of Computer Science at Dalhousie University. He received a diploma in Electrical Engineering from the National Technical University of Athens, Greece, in 1980 and M.Sc. and Ph.D. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute

of Technology, Cambridge, Massachusetts in 1986. At Dalhousie, he served as Director of the Graduate Program (1999-2002) and he is currently Associate Dean, Research. His current research activity is centered on modelling and mining of content and link structure of Networked Information Spaces.

**Nur Zincir-Heywood** is Associate Professor in the Faculty of Computer Science at Dalhousie University, NS, Canada. She obtained her B.Sc., M.Sc. and Ph.D. in Computer Engineering from Ege University, Turkey in 1991, 1993 and 1998 respectively. Her research interests include network services and management, network information retrieval, and the effects of the Internet and information technologies on socio-economic development.

**Markus Latzel** is President & CEO of Palomino System Innovations Inc, a successful university spin-off in its seventh successful year. He received his M.Eng. in Computer Science and Electrical Engineering from University of Paderborn, Germany in 1999. His research and publications are in areas of intelligent systems, robotic systems, knowledge management, hierarchical databases, HCI and computer vision.

**Sotirios Stergiopoulos** received his B.Sc. in Computer Science from York University, Toronto, ON, Canada. He was previously with MacDonald Dettwiler and Array Systems Computing. He is currently Lead Application Developer for Palomino System Innovations, Inc., Toronto.

## References

- [1] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
- [2] Ralph Bergmann. *Experience Management: Foundations, Development Methodology, and Internet-Based Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.
- [3] Valeria Cardellini, Emiliano Casalicchio, Michele Colajanni, and Philip S. Yu. The state of the art in locally distributed web-server systems. *ACM Comput. Surv.*, 34(2):263–311, 2002.
- [4] R.N. Cronk, P.H. Callahan, and L. Bernstein. Rule-based expert systems for network management and operations: an introduction. *Network, IEEE*, 2(5):7–21, Sep 1988.

- [5] David A. Grossman and Ophir Frieder. *Information Retrieval: Algorithms and Heuristics, Second Edition*. Springer, Dordrecht, The Netherlands, 2004.
- [6] H. Inamura, O. Takahashi, T. Ishikawa, H. Shigeno, and K. Okada. Automating detection of faults in tcp implementations. *Advanced Information Networking and Applications, 2004. AINA 2004. 18th International Conference on*, 1:315–320 Vol.1, 2004.
- [7] L. Lewis. A case-based reasoning approach to the management of faults in communication networks. *INFOCOM '93. Proceedings. Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future. IEEE*, pages 1422–1429 vol.3, 1993.
- [8] Mirjam Minor. *Erfahrungsmanagement mit fallbasierten Assistenzsystemen (Experience Management with Case-based assistant systems)*. PhD thesis, 2006.
- [9] Mirjam Minor and Christina Biermann. Case acquisition and semantic cross-linking for case-based experience management systems. In Du Zhang, Taghi M. Khoshgoftaar, and Mei-Ling Shyu, editors, *IRI*, pages 433–438. IEEE Systems, Man, and Cybernetics Society, 2005.
- [10] G. Penido, J.M. Nogueira, and C. Machado. An automatic fault diagnosis and correction system for telecommunications management. *Integrated Network Management, 1999. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on*, pages 777–791, 1999.
- [11] P. Ramnarayan, N. Cronje, R. Brown, R. Negus, B. Coode, P. Moss, T. Hassan, W. Hamer, and J. Britto. Validation of a diagnostic reminder system in emergency medicine: a multi-centre study. *Emergency Medicine Journal*, pages 619–624, 2007.
- [12] P. Ramnarayan, G. Kulkarni, A. Tomlinson, and J. Britto. Isabel: a novel internet-delivered clinical decision support system. *Current Perspectives in Healthcare Computing*, pages 245–246, 2004.
- [13] P. Ramnarayan, A. Winrow, M. Coren, V. Nanduri, R. Buchdahl, B. Jacobs, H. Fisher, P.M. Taylor, J.C. Wyatt, and J. Britto. Diagnostic omission errors in acute paediatric practice: impact of a reminder system on decision-making. *Medical Informatics and Decision Making*, 2006.
- [14] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.
- [15] J. Stearley. Towards informatic analysis of syslogs. *Cluster Computing, 2004 IEEE International Conference on*, pages 309–318, 20–23 Sept. 2004.
- [16] Neal J. Thomas, Padmanabhan Ramnarayan, Michael J. Bell, Prabhat Maheshwari, Shaun Wilson, Emily B. Nazarian, Lorri M. Phipps, David C. Stockwell, Michael Engel, Frank A. Maffei, Harish G. Vyas, and Joseph Britto. An international assessment of a web-based diagnostic tool in critically ill children. *Technology and Health Care*, pages 103–110, 2008.
- [17] R. Vaarandi. A data clustering algorithm for mining patterns from event logs. *IP Operations and Management, 2003. (IPOM 2003). 3rd IEEE Workshop on*, pages 119–126, 1–3 Oct. 2003.