Spring 2009

Open Source Trouble Ticket System

Steve Chapman
Regis University

OPEN SOURCE TROUBLE TICKET SYSTEM

by

Steve Chapman
chapm032@regis.edu

A Thesis/Practicum Report submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Information Systems

School of Computer and Information Sciences
Regis University
Denver, Colorado

February 8, 2009

Abstract

The productivity of many organizations in the IT industry suffer greatly because most have yet to implement a process to manage and track IT issues from the time a problem is reported by the end-user to the time in which the problem is resolved. Until such a process exists, many organizations will find themselves unable to effectively manage time and resources thus resulting in unnecessary downtime, decrease in productivity and ultimately becoming an organization who "throw money at the problem" instead of fixing the root cause. One such solution would be to use a trouble ticket system application. There are many trouble ticket system applications developed by commercial companies but for organizations with limited financial resources there are just as many open source, or free, solutions to consider. This paper discusses the various phases used to implement a zero cost open source trouble ticket system application called Open Source Ticket Request System (OTRS) in a government work environment.

Acknowledgements

Place any acknowledgements here. Your acknowledgements should be in good taste and should not exceed one page in length.

Table of Contents

List of Figures

List of Tables

Executive Summary

The goal of the project was devoted to addressing two major problems that involved two engineering groups. The first group, consisting of software engineers, needed a standardized process to create and track all hardware and software related issues in the form of trouble tickets. The second group, consisting of system engineers needed a process that could manage and track those trouble tickets until they were resolved. What both groups had in common was the need for a trouble ticket system application.

The second problem, which is the reason behind the first problem, was not having available funds to invest in a commercial off the shelf (COTS) solution. In an effort to mitigate both problems, research was performed for several weeks and as a result Open Source Ticket Request System (OTRS) surfaced as a robust trouble ticket system application that was free to own and operate.

During the week of April 30, 2007 OTRS was implemented and turned over to the software engineering group who begin using the application almost immediately. Each trouble ticket submitted was automatically routed to the system engineering group who was responsible for working the ticket until resolved. Once the reported problem was resolved, the ticket was closed and an automated message was generated and sent back to the requestor. Prior to this effort, the system engineering group was overwhelmed with make-shift lab requests written on sticky notes and left in various locations such as doors or desks, but since the rollout of OTRS, there has been 100% participation using the new system.

Chapter 1 – Introduction

Those who rely on technology in any capacity have experienced the inconvenience caused by its malfunction. These malfunctions are more pressing when encountered in the work environment because they impede work which stifles productivity. Sometimes these malfunctions are as simple as a minor hardware failure requiring a quick fix, to more complicated and urgent matters requiring more time and skill to resolve. Managing and tracking large numbers of hardware, software, or other IT related issues that are reported daily can be an overwhelming and challenging task for any IT department. Not having the funds to implement such an application only compounds the problem thus requiring an alternative solution. This project involves implementing a trouble ticket system application in a government environment that is free to use and operate.

1.1- Review of Existing Situation

Lockheed Martin is an engineering firm dedicated to providing engineering expertise to commercial and government organizations. Their most recent undertaking is a project in which they provide custom applications that can be used to help protect government networks against cyber attacks. There were several groups assigned to this effort but the two engineering groups chosen by Lockheed Martin to lead the efforts were the Software Engineering group and System Engineering group. These two groups were chosen because their responsibilities complement each other. The software engineering group's primary responsibility was to develop custom applications and ensure those applications test successfully in a lab environment. The system engineering group provided more of a support role by making sure the lab had all the required hardware and software needed to test each application build.

When the two groups begin working together the process used to submit lab requests was very informal. Often times a verbal request or a request in the form of an email or sticky note was the method used by the software engineering group to bring request lab support from the system engineering group. This informal process was manageable when lab requests were few but quickly proved inadequate as the amount of requests increased. It became difficult for the system engineering group to keep pace and as a result, lab requests were not addressed in a timely manner. The informal process no longer worked and a more standardized process was needed

.

Table1: The following table shows the average time required to resolve each lab request using the informal process.

# of Trouble Tickets	**Average Time to Resolve**
1	1 Hour
2	2 Hours
3	3 Hours
6 or more	6+ Hours

The table above shows the average time required to resolve each lab request using the informal process. As the workload increased the ability to handle each request in a timely manner became more challenging. Each lab request required on average an hour to resolve and an entire day to resolve eight or more! When issues were not addressed and resolved in a timely manner, usually an hour or less, it only created a bigger wedge between the two groups as the software engineering group began to feel ignored. Quite the contrary as the system engineering group were simply overloaded!

The quality of the applications, or lack thereof, was quickly pointed out by the Quality Assurance (QA) group as well as by various end users who depended on these applications to keep their networks safe and secure. A long-term solution was definitely needed to manage the opening, tracking, and closing of lab request tickets. A higher quality product delivered in a more efficient manner was to be expected.

The management staff welcomed the idea of a more efficient process but their already stretched budget didn't allow for the purchase of a commercial off the shelf (COTS) solution. The solution to both shortcomings was a trouble ticket system application that is free to use, operate, and distribute. Streamlining the process to open, track, and resolve IT issue as well as implementing new capabilities was an endeavor worth pursuing.

1.2 - Project Goal

The goal of the project was to implement a trouble ticket system application inside a government environment without compromising security. The application provided a simple solution to create and track trouble ticket request as well as a process to create custom queues used to manage and organize those requests. The caveat to the overall project goal was the unavailability of funds to support such an effort. Therefore, the second goal of the project was to find a zero cost solution.

One of the key benefits of implement a process is the ability to more efficiently manage time and resources. The time spent on each issue substantially decrease because each issue was be prioritized and addressed based on the urgency and not on a first come first serve basis. Urgent issues were flagged accordingly and addressed immediately while less urgent issues were addressed last but still within a reasonable amount of time.

An additional bonus for using a trouble ticket system application is the central repository or knowledge database where all resolved issues are kept. Similar issues that occurred were searched in the database and resolved resulting in a quicker resolution time and fewer man hours spent on each issue.

1.3 - Barriers and/or Limitations

Although using an open source trouble ticket system application provided an immediate impact to the two groups, such a system could not be implemented without the expectation of many obstacles along the way. Several barriers and limitations were expected at the start of the project while others were discovered as the project progressed.

The first barrier encountered was meeting with the Information Systems Security Officer (ISSO) to seek permission to install an open source application developed by the public in a sensitive government environment. The ISSO is ultimate responsible for what goes on the network excluding the external (unclassified). The project could not go any further until this barrier was satisfied. Meeting with the ISSO proved to be a major hurdle and direction changer due to their strict guidelines for all software used on the internal network. The requirements set forth by security to receive approval is that the software must be developed and owned by a company residing in the United States and the application must be free from any known security vulnerabilities. Needless to say, the request to use an open source application on the internal network was dismissed and the reason cited was the unwillingness to compromise security and the integrity of data by using an open source application. As a workaround solution, the open source trouble ticket system application was designated as “unclassified” and placed on the external network thus abandoning the original plan. The new direction also resulted in the addition of a new policy which stated, *“Sensitive information could not be entered in the ticket”*.

A second limitation encountered was implementing an application without adequate technical support. Most open source applications are developed by the community which means they do not have a traditional “technical support” department to call for assistance with installation, configuration, or application usage. Instead, most questions for open source applications are posted online in an open forum and answered by other users. As a result, the end user may be forced to wait for an answer resulting in a loss of productivity or because of impatience, application misuse. Asking the software engineering group to potentially wait an indefinite amount time was unacceptable. Especially in an agile environment that demands immediate answer. There did not appear to be a solution to completely mitigate this limitation because it was the tradeoff of using an open source product. However, the proactive approach taken to minimize potential downtime was learning as much about the application as possible and using the expertise of the software engineering group as an in-house resource with software related issues.

A third and final limitation encountered was the absence of a disaster recovery plan such as a backup server. Accepting this limitation was more of a choice than a limitation because the main objective of the initial rollout was to keep hardware requirements at a minimum while placing more focus on functionality, documentation, and most importantly time to implementation. The absence of a backup server created a single point of failure which was a risky approach to take but the two weeks given to implement a solution did not provide enough time or resources to implement redundancy. As a pressed for time alternative, the hard drives used in the hosting server were mirrored using Disk Management, a feature found in Microsoft Window's Server 2003 operating system. Mirroring works by writing data to one hard drive and synchronizing with another hard drive thus creating two separate but identical hard drives. If one hard drive fails, the other hard drive has a mirror copy of the data which can be used to quickly stand up another server.

These barriers and limitations were important enough to be noted however, the benefits to moving forward with the project far outweighed the drawbacks.

1.4 - Project Scope

The scope of the project was to implement a zero cost trouble ticket system application that was used by the software engineering group to create and submit trouble tickets for all of their hardware and software related issues. All trouble ticket request created were routed to the system engineering group who were responsible for resolving those issues. The trouble ticket system also provided the capability to track the status and work history of each ticket until closed or resolved.

The project was comprised of four phases: research of the problem and potential solutions, procurement of hardware and software, system build, test and integration. The project included researching existing open source trouble ticket system applications with a greater emphasis on functionality, ease of use, and the ability to personalize to suit the needs of both engineering groups. Applications that were too complicated would have caused more harm than good by requiring additional training and assistance which was not in the budget. Applications that were overly simplified would not have the ability to accommodate future growth of the teams and their subsequent needs.

Phase two compared the hardware requirements of each application to eliminate those that could not be supported by the hardware currently available on hand. The minimum hardware requirements for each open source application considered were gathered and compared to determine whether the application could be accommodated based on the hardware equipment on hand. Those that could not were eliminated from contention. Lockheed Martin was already a Dell preferred organization so reliability was not much of an issue as they were already comfortable with Dell's products and customer support. An added piece of mind was a comprehensive maintenance agreement already in place. Only hardware that met or exceeded the minimum needs of the application were considered.

The next phase of the project entailed reviewing the underlining programming language for each application considered and comparing those technologies with the skills matrix of the software engineering team. Regardless of the application chosen, the unique environment required that a group logo, government warning banner, and several other unmentionables be applied to the front end user interface. Therefore, it was important to ensure that the programming language did not require a steep learning curve and there were adequate skills on hand to make the necessary changes.

From a security standpoint, there were many important aspects to consider. In order for any application to be implemented it had to be approved by the ISSO who required that each application pass a rigorous security checklist. The main concern of the ISSO was to ensure that all security guidelines set forth by the government had been met or exceeded. Being asked to weigh in on an open source solution was a first for the ISSO team and as a result many concerns were raised. They were not convinced that an open source solution provided the same level of quality control and security checks and balances as their COTS counterparts. Cowan sees it differently in his book titled, *Software security for Open-Source Systems*. He writes, “Open source code is widely considered to be highly effective for mission-critical functions, precisely because its code can be publicly scrutinized for security defects. Any defects found can be immediately addressed as open source provides users with the ability to security-enhance their own systems rather than being locked into a system purchased from a proprietary vendor”. (Cowan, 2003).

The information obtained up to this point was used to procure the hardware necessary to host the application. Fortunately, the individual pieces of hardware needed to build the platform were not application specific therefore a server with adequate memory, hard drive space, and cpu was all that was required. All of the applications considered were compatible with Microsoft’s Server 2003 which was a baseline operating system.

The target for the initial rollout of this application was set for April 30, 2007 which was exactly two weeks to complete the project. The date was firm therefore advanced features of the application could not be explored during the initial rollout.

Chapter 2 – Review of Literature and Research

There are many solutions available to satisfy the requirements described in Chapter one. The available solutions range from commercial off the shelf products that require the purchase of a user license to products that are free to use and distribute. The literature portion of this paper cannot dissect in detail every solution considered. Instead, the research focuses on the technology behind trouble ticket system applications and the advantages of using an open source solution.

Lytras & Naeve, (2007), said it best, "all the research of open source applications is for naught if implementation is not built around a solid evaluation of core business criteria in all their complexity". (Evaluating Open Source in Government section, para. 8). This statement is precisely the goal of the project paper which is to evaluate core business needs and present a solution that is both practical and attainable.

With this goal in mind, the outline of this paper is as follow. In chapter two I begin by defining trouble ticket systems and open source applications. I explain the advantages of Open Source applications and how such an application is typically used in an IT environment. The table in chapter two lists several open source solutions that were considered for this project along with a brief review to include the technologies involved.

In chapter three I discuss the methodology used for research and the outcome. I close the chapter by introducing Open Source Request System (OTRS), the application selected for the project. Chapter four addresses the history of the project — how it came into existence, how the project was managed, significant events and changes that occurred during the research of the project, whether the project met the goals, what went right or wrong during the project, and analysis of results presented in the paper. Finally, Chapter five discusses lessons learned, what could have been done differently, where the project is progressing from this point and a concluding statement to the paper.

2.1 - Trouble Ticket System Defined

A trouble ticket system as defined by Zazachat.com, is a software application that organizations use to keep track of problems, work flow, resolution, and expended time. This type of application might also be referred to as an issue tracking system or an incident ticket system and is essentially a computer software package that manages, and maintains lists of issues as needed by an organization (Zazachat, para. 1).

A trouble ticket system application is a mechanism used in an organization to track the detection, reporting, and resolution of some type of problem. Trouble ticketing systems originated in manufacturing as a paper-based reporting system but are now mostly web based and associated with customer relationship management (CRM) environments, such as call centers or e-business web sites, or with high-level technology environments such as network operations centers (NOCs). As a ticket moves though the system, it is usually classified as a certain type of problem which in turn determines the skill set or expertise required to resolve the problem. Until the problem is resolved, the trouble ticket remains in the work queue with problems of highest priority taking precedence in terms of work flow.

Since trouble ticket system applications store large amounts of proprietary data in the form of a centralized repository, security is a major concern. This information should be protected at all times because the information may be used to expose internal vulnerabilities and security weaknesses if placed in the wrong hands. For example, internal users may report an ongoing operating system problem which may suggest to an onlooker that the proper patches and updates have not been applied thus exposing potential “attack points” in the system.

In Request for Comment (RFC) 1297, The Internet Engineering Task Force’s Network Working Group provided the best analogy of how a trouble ticket system works. RFC’s are memorandums issued by the Internet Engineering Task Force (IETF) that describe the workings of the internet and internet connected systems. A trouble ticket was compared to a patient’s hospital chart. Both define a problem and request the help of several different people or organizations to determine a resolution. A patient (trouble ticket) can not leave the hospital (work queue) until the problem has been resolved (ticket closed).

2.2 - Open Source Defined

The most impacting pre-open source movement was the development of telecommunication protocols that are still in existence today. They were developed by researchers who used Request for Comments (RFC). Described as forward thinking at the time, these collaborative efforts eventually lead to the birth of the Internet in 1969. There are other instances of open source movements such as IBM’s free release of their operating system in the 1960’s and the user groups that formed to assist in the exchange of software.

During a strategy session held at Palo Alto, California, the decision by some people in the free software movement to use the term “open source” instead of free software was born. This decision was the result of Netscape's January 1998 announcement that they intended to release the source code for their Navigator browser.

The group of individuals at the session included Christine Peterson who came up with the name “open source”, Todd Anderson, Larry Augustin, Jon Hall, Sam Ockman, Michael Tiemann and Eric S. Raymond. These individuals seized the opportunity before the release of Netscape’s Navigator source code to make the case to change the name from the often misunderstood term free software. Netscape went on to license and release their source code as open source under the Netscape Public License and subsequently under the Mozilla Public License (Muffatto, 2006). Software developers now use the term open source to describe software whose source code is available to the public.

Most would say the biggest advantage of open source software is that open source software is made available free or at a reduced cost. This generalization is not exclusive to open source software because several commercial software products are also free. Take for instance many of the popular internet browsers. Microsoft’s Internet Explorer is a good example of free software that is not considered open source but is free to use. According to Sharma (2002), open source applications provide developers with a sense of loyalty and empowerment due to a sense of ownership and attachment to the end product (Sharma, 2002). This sense of ownership motivates developers to remain an active contributor to the effort.

Additionally, open source software does not require the same marketing and logistical services as commercial based software. Many of these services include but are not limited to design and print, fulfillment and distribution, packaging production, and advertising. The open source development approach affords more flexible technology and quicker innovation by becoming modular thus allowing developers to build custom interfaces that are more flexible and scalable.

Commercial software manufacturers' claim that the open source development processes lack the checks and balances usually found with COTS products. Their assertion is that more defined stages in the development process such as system testing and documentation are ignored. Their claim is more widespread in smaller projects where one or two programmers are involved. However, larger more popular projects tend to enforce rules that promote team work and provide a more polished end product.

The biggest difference between open source software and software available without a fee is a combination of characteristics which are described below. All of these characteristics combined create a synergistic impact which encompasses the real advantages of the open source model.

2.3 - Open Source Advantages

Open source systems and applications often appear to offer significant benefits over proprietary systems. Consider some of the metrics they compete on. First of all, open source products are usually free of direct cost. They are often superior in terms of portability. Source code can be modified because you can see it and modifications are allowed by the licensing requirements. The products may arguably be both more secure and more reliable than systems developed in a proprietary environment. Open source products also often offer hardware advantages, with frequently leaner platform requirements and newer versions can be updated for free. (Deek & McHugh, 2008)

The benefits mentioned by Deek & McHugh was realized in this project and also several additional benefits to open source applications such as flexibility, multiple uses, expansive rights, competitive advantage, and reliability will be expounded upon below.

2.1.1 - Lower Cost

The most obvious and compelling reason to use an open source solution is the initial lower or zero cost of ownership. Organizations and users are free to user, copy and distribute open source software without incurring any fees. Consider an application with an installed base of 100 users and a 10-person development team using a $500 licensed commercial product. This would total $55,000 in expenses. Now consider the use of a competing Open Source product. The organization could immediately eliminate the large expense and increase the install base without incurring additional expenses. Other financial benefits can be realized as well. Because Open Source is free to copy, the expense of license management isn't incurred. In addition, legal departments only have to review and approve an Open Source license once for all projects using that license rather than each time for each commercial product license. Using popular Open Source projects can reduce training expenses by providing a larger resource pool. Developers can be hired from outside the company with existing knowledge of Open Source frameworks or projects. It's often difficult to hire developers that have knowledge of a proprietary commercial framework. (Judd & Bodden, 2004)

2.1.2 - Flexibility

In the words of Linus Torvalds Ghosh, 1998b), “In fact, one of the whole ideas with free software is not so much the price thing and not having to pay cash for it, but the fact that with free software you aren’t tied to any one commercial vendor. Making source code available to the general public allows for the unlimited tweaking and improvements to the software product. Developers are afforded the flexibility to use the source code to build a better more flexible application. One of the problems with commercial software applications is the risk of compatibility issues when mixed with applications made by different vendors. Many commercial solutions do not have the flexibility of being installed on multiple operating systems therefore the user is often tied to a particular system or vendor.

2.1.3 – Community Support

With open source software, users have at their disposal a highly motivated community of support willing to answer questions (Lakhani and von Hippel 2003). Linux, a popular open source technology, have numerous Linux User Groups (or LUGs) that do an excellent job providing service. (Feller, Fitzgerald, Hissam & Lakhani 2005) Using a COTS product require users to contact the vendor for technical support if a problem is encountered. In many cases, the level of support is poor (especially in the case of free service) or the user may have to pay a fee for premium technical support. Incurring a fee for each time technical support is needed is not a viable option for many organizations.

2.1.4 - Expansive Rights

Expansive rights of open source software give users the freedom to redistribute, modify, and improve upon the source code without the liability of violating intellectual property laws. Having full rights allow open source software to be modified by a larger community developers whose main goal is to improve the product and eliminate bugs. These rights are the major difference between open source software and nearly free software. Unlike COTS products, the universal right to redistribute open source software cannot be revoked which makes it more attractive for developers to lend their expertise as they feel a sense of ownership.

2.1.5 – Agility

The evolution of open source applications is more agile than their commercial offerings because of their shorter release cycles and expedited development time. If for no other reason than the fact that most open source applications provide nightly snapshots or direct access to the source code repository allowing applications to be developed on 24/7 basis. As a result, organizations do not have to wait for a vendor's next release to get a bug fixed because having the source code available provides a means for the organization to fix the bug itself. Organizations willing to contribute to Open Source projects can also have influence on the future direction of the project. Unlike proprietary development, Open Source has the advantage of being reviewed and tested by potentially hundreds or thousands of users. (Judd & Bodden, 2004)

2.1.6 – Industry Support

Industry support is another advantage to open source applications. Many major companies such as IBM, Sun, Oracle, BEA, and Borland are using Open Source projects. Therefore, these organizations have a vested interest in the project's success because their products rely on it. Contributors to the Java Open Source projects aren't necessarily the independent programmers writing code in their spare time anymore. Many of these large companies have departments dedicated to Open Source. In addition, many of the Open Source projects such as Eclipse, NetBeans, and Tomcat were initially donated by large corporate organizations and progressed forward by the community. Consider the use of Open Source as a means of expanding an organization’s development team to include some of the best developers and corporate backers from all around the world. Access to the source is an important advantage of Open Source. The source code is the only 100 percent–accurate documentation. (Judd & Bodden, 2004)

2.1.7 –Security

Many developers cite transparency as the main reason open source applications are more secure than their proprietary alternatives. According to Andrew Bardin Williams, if customers and developers can look at the code, they are more likely to find a bug and create a patch. In a closed source model, customers must rely on the vendor to identify, diagnose and issue a patch, which can be a lengthy process. He went on to say, Government agencies using open source also benefit from a broad user community in the commercial space that is committed to maintaining security. These user communities are involved with testing the software, developing fixes and sharing patches.

OpenSolaris.org is an example of an active user community. They currently have 11,000 members with 1,000 being actual Sun employees. When a security flaw is made known, thousands of users simultaneously work to find a solution. "Government agencies using the same software platform can take advantage of these resources rather than developing their own patches or relying on vendors. Once a patch is developed, usually the open source vendor agrees to support it and incorporate it into subsequent releases". (Security Benefits of Open Source section, para. 3).

2.4 - Considered Applications

One of the most contentious issues in the literature has been the relative virtues of the open source and proprietary development process. Advocates of open source software have long claimed that the open source development process leads to superior software (Raymond, 1999).

In order to determine the most effective open source trouble ticketing system application to utilize for the project, a variety of application were evaluated. The applications evaluated were not only comprised of the latest and greatest but also a mix bag of older applications as they could serve as the base or skeleton for in-house modifications. Table 1 list several trouble ticket system applications that were considered followed by a brief description of the application and why it was not an ideal fit for the project.

Table2: A list of trouble ticket system applications that were considered.

Name of Application	**Web Address**
PHP Support Tickets	www.phpsupporttickets.com
OS Ticket	www.osticket.com
TicketSmith	www.osticket.com
Help Desk Lite	http://www.helpdesklite.com/
Cerberus	http://www.cerberusweb.com/
Support Trio	http://activecampaign.com/supporttrio/index.php
Kayako SupportSuite	http://www.kayako.com/
BATTS	http://www.xisp.net/batts/
SitePanel3	http://sitepanel3.com/

PHP Support Tickets

PHP Support Tickets is an application written in PHP5 and utilizes MySQL database. The application will not operate unless both of these applications are present.

The installation is a simple process that can be completed in approximately five minutes. PHP Support Tickets is broken up into three user levels: Administrators, Moderators, and Clients. PHP Support Tickets come in two versions, free and premium. This application was ruled out because the free version does not include the latest features or bug fixes. These items are only included with the free version.

OS Ticket

The core components of OS Ticket are Perl gateway, MySQL database, and PHP. The core features are the ability to create categories or departments, create representatives or supporters and define groups. OS Ticket has the ability to process an unlimited amount of emails that are sent between administrators, supporters, and users. OSTicket does not seem to have as many users and online support groups as the other applications researched.

TicketSmith

Ticketsmith is marketed as an all-in-one web-based email support system. The basic functionality of the application is messages are sent to the support email list, cataloged, and inserted into a database for easy viewing on the web. Replies are processed the same way. TicketSmith offers fast sorting and search capabilities and email notifications of new tickets. This application does not support HTML emails which would not be popular among users who choose this format when submitting and viewing ticket requests. At the moment of writing this paper, TicketSmith is no longer supported therefore bug are no longer addressed and the application is offered on an as is basis.

Cereberus

Cereberus is a popular ticket system application that provides a host of advanced feature such as Customizable Ticket Views, Adaptive Anti-Spam, Custom Field Groups, Reporting System and Email Templates. It also provides an integrated knowledge base that puts proven solutions in the hands of even the newest team members.

The ticket display screen is a one stop shop for issues including a customer's past support history, a log of actions performed on the ticket and tracking of custom data. These features are what makes Cereberus stand out above the rest. An added benefit to Cereberus is the included spam and anti-virus feature that can even adapt to the latest tricks. The free version allows full functionality but is limited to a single email address which would not be adequate for my needs.

Support Trio

Support Trio is an integrated solution whose primary focus is on managing and controlling support traffic. The application includes a number of advanced features such as spell check, spam filter, custom reports, and integrated knowledge base. Users and support staff can easily update tickets through a centralize location rather than have tickets spread out in multiple places. Tickets are archived and can be accessed at a later date. A standout feature is the ability for a user and client to communicate about a ticket using their own email address.

Unlike other ticketing systems, Support Trio allow administrators to decide what requirements must be met before a user can open a ticket. An administrator may require that every user of the system go through a registration process prior to using the application or registration can remain optional. Users who do wish to register will not have access to advanced features and capabilities that are afforded to registered users. The downside to SupportTrio is the free version has many limitations such as a maximum of three administrator accounts, three departments and one parsed email address.

Kayako SupportSuite

Kayako SupportSuite is a fee based product that packages many of the most sought out features of trouble ticket system technology into one application. One of the standout features includes AJAX based Rich User Interface which displays real-time results from the knowledgebase to end-users before they submit their issues. ModernBill, considered the leader in automation technology, is another advanced feature included with this product. There is also a feature called Viewshare and Teamwork. Viewshare allow users to guide their clients in real time, voice chats, work schedules and escalations and the Teamwork module allows them to create shared Events, Contacts and Tasks. This application started out as open source but now offers only a 7 day free trial or full version starting at$29.95

BATTS

BATTS is an open source trouble ticket system application that is used from a Linux command line. The open source community website is still up and running with a copy of the install. However, the page says last modified on September 6, 2003 which suggests to me that the development community has long abandoned this effort.

SitePanel3

Although SitePanel3 was designed to be user friendly, the application packs a host of features for the experienced user allowing them to fully leverage the range and features the system has to offer. SitePanel3 is easy to install and doesn't require major configuration prior to becoming operational. It is possible to have the helpdesk application up and running in as little as 5 minutes. The system utilizes common server software and PHP modules to ensure proper operation on any web server.

SitePanel3's configuration screen allows the user to easily make changes to how the system operate from configuring email to creating custom fields and departments. The user interface can be customized and different email templates can be defined complete with custom variables, for each ticket department. SitePanel3 also can be customized to the end-user's interface allowing the color scheme, logo, and look and feel to be modified. The application is feature packed and considered too complex for a quick turnaround solution.

2.5 - Application Choice

Choosing the right application that would bridge the communication gap on the project without posing any significant drawbacks or security risks was by no means an easy task. There were many pros and cons with almost all of the applications evaluated. Some applications provided many advanced features but was too complex to install while others had a simple installation procedures but didn't include desirable features such as

HTML email support or queue based ticket management. Then there were others that started out as open source applications but have since changed to fee based. This bait and switch tactic is considered by many in the IT industry as a great marketing ploy. An application starts out as open source, gains notoriety and clientele from those who depend on it, and then switches to commercial or fee base.

This project was approached with realistic expectations therefore regardless of the application chosen, some modification would be necessary. Selecting an application written in a programming language that could be easily modified by someone not versed in a programming language was ideal. Taking everything into consideration, Open Ticket Request System (OTRS) was the best choice. The primary reason for selecting OTRS was its ease of installation, simple setup, minimum hardware requirements, extended knowledge and user base, and programming language. Many of these features will be explained further later on in the paper.

2.6 - Open Ticket Request System (OTRS)

OTRS is a web based application that can be installed on a Microsoft Windows or Linux based operating system. The application made it possible for the software engineering group to open, update, and track trouble tickets via the web using a username and password. Ticket requests could be “locked” or acted upon based on individual, teams, or category queues as well as a particular customer. The major components of OTRS are MYSQL database, Apache web server, and SendMail application which were all rolled up into one easy to install executable. To determine which operating system platform provided the easiest install, fastest response times and overall ease of operation, OTRS was installed on Microsoft’s Sever 2003 and Linux Enterprise Server operating systems using identical hardware. Each operating system performed well but it was determined that Microsoft's Server 2003 was the operating system of choice because of its seamless integration, familiar "point and click" graphical user interface, and all in one executable.

A strong characteristic and decision factor of this application was its queue based management of requests. This feature automatically routed incoming tickets into their appropriate queues without manual intervention. This feature helped streamline the process because as tickets are entered into the system they were automatically routed to their appropriate queues. Several queues used during setup were hardware, software, telephony, and miscellaneous. The hardware queue was used for any requests related to hardware (ie. servers, workstations, hard drive storage, etc.). The software queue captured all software requests to include software installs, user account creation, and/or configuration changes. When new software developers join the team a request is created and routed to the telephony queue to have their telephone account setup. The Miscellaneous queue was created to capture all requests that did not fit in the other queues.

Another strong feature of this application is the ability to create filters based on various characteristics of incoming emails. Filters were used to organize and prioritize incoming emails based on specific criteria such as subject matter or sender. Email templates were created for common requests. Below is a list of more specific characteristics regarding why this application was chosen.

2.6.1 - OTRS Basics

OTRS account types are limited to three distinct groups: customers, agents and administrators. A "customer" account type is giving to anyone who is only authorized to access the system for the purpose of reporting a problem (ie. opening a ticket), updating a ticket, or checking the status of an existing ticket. This account type was suitable for the software engineering group. Customer account holders do not have rights to change the status of a ticket or its properties.

Agents are anyone who has authorization to change the status of a ticket or its data. The system engineering group was assigned agent accounts thus given full access to change ticket properties such as priority, queue, and responsible agent. Agents can also close tickets prematurely or once the problem has been resolved.

Administrators are the third and final account type. They have full rights to modify tickets to include its associated data. Administrators can also make system configuration changes such as create and delete queues and add/modify/delete user accounts and their rights. The three OTRS user accounts are created by default but are not mandatory. All three can be configured with detailed granularity when assigning rights and permissions.

2.6.2 - Tickets

Tickets were created by accessing OTRS via a web interface or using SNMP which allowed the opening of tickets using an external email address. Many of the open source trouble ticket system applications considered were close to satisfying the needs of the group but the interfaces were too busy and the process required to create tickets were too cumbersome. One application in particular required more than eleven fields to be populated before a ticket could be created. Only two fields are required to open a ticket in OTRS, a brief description of the problem and the queue that best suits the problem being reported.

Each open ticket in OTRS maintains a complete work history but only if the ticket has been properly updated with the latest information. Tickets can be updated manually or through an automated process. Tickets can also be split into multiple cases and several tickets relating to the same case can be merged into one.

2.6.3 - Queues

OTRS use queues to separate tickets into smaller groups based on the problem reported or person assigned to solve the problem. The four queues created for this project were hardware, software, telephony, and miscellaneous. Incoming tickets were automatically routed to the appropriate queue.

2.6.4 - Email

One feature that made my life easier is the ability to create outgoing emails from templates consisting of an introduction, signature and main body. Custom templates can be created for individual queues or to address a specific problem being reported. On occasions my department will have a major issue that may take an entire business day to resolve. In those cases, a custom template can be created and sent out with the status of the outage. The biggest drawback to this feature is that templates can be used for replies but not forwards. Forwarding of incidents is an important feature of any engineering team as information often needs to be disseminated along to management or others who do not have access to the OTRS application.

Chapter 3 – Project Methodology

3.1 - Project Life-Cycle

Systems Development Life Cycle (SDLC) or sometimes just (SLC) is commonly referred to as a software development process but can also refer to a distinct process independent of software. SDLC is also known as information systems development or application development. SDLC is a systematic approach to problem solving and is composed of several phases, each comprised of multiple steps:

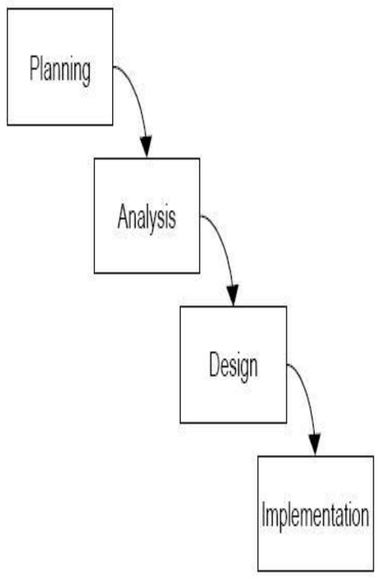


Figure1: A simple waterfall model.

3.2 - Project Planning & Requirements Gathering

This step outlines the plans for creating the system by defining the high level requirements of the system, the details and plans for developing the system, and managing and monitoring the project until completion. The research group is comprised of thirty Software Developers and three System Engineers. The System Engineers are responsible for supporting the needs of the software engineers which may include hardware, software, network connectivity, or a combination of all three.

The current communication method utilized by developers whenever they have an issue is to paste sticky notes on our desk or verbally express their issues in passing. This nonstandard method has proven ineffective because issues are often forgotten or urgent issues that should have received immediate attention are not moved to the top of the workload. The software development team is growing exponentially and the need for a more standardized process is paramount. My expectations for a trouble ticket system application was centered on security, ease of use and maintenance. Users of the application will be broken down into three groups: Customers, Agents, and Administrators.

Each software developer would need a customer account which allowed them to log onto the system via the Customer Login page using the supplied username and password. After logging onto the system a new trouble ticket could be created or the status of an existing trouble ticket could be gathered. Each trouble ticket request should provide a field to enter a brief description of the problem being reported along with the requestor's contact information. After a trouble ticket request is submitted, the requestor should have the ability to logon to the system to check the status of the request, update the request, or create a new request. Customer account holders should not be able to modify a ticket or its properties in any way.

The Systems Engineering group will use Agent accounts. Agents resolve trouble ticket request submitted by customers (software developers). Agents should be able to create and delete user accounts, user groups, and queues as well as modify rights and permissions. An agent account should also have the ability to modify trouble ticket request properties to include update work history, change ticket status, reassign to another queue or close tickets. Agents can also open tickets on behalf of customers who phone in their trouble ticket.

The last OTRS account to discuss is Administrator accounts. Anyone with an administrator account will have all rights and privileges to make system wide changes. Administrators should have access to configure POP and SMTP settings configurations as well as create and delete queues and add/modify/delete user accounts and their rights.

3.2.1 - Analysis

This step brings users and engineers together to formalize and analyze the business requirements. In order to choose the best trouble ticket system application it was necessary to evaluate each application extensively. The main source used for research and evaluation was the internet as it proved to be an invaluable tool for understanding open source ticketing system applications and the viable choices that were available.

In an effort to gain a better understanding of how these applications work, user groups and online forums were frequented to better understand what others who have hands-on experience using many of these open source ticket system applications in their environment had to say. Additionally, several meetings were held with the software development team to determine what their needs were for a trouble ticket system application. After convincing the software development team to be open minded to a new business process, their needs were quite simple. They all agreed that an easy to use application that offered a single sign on and a simple and quick process to open and close trouble tickets was ideal.

3.2.2 - Design

The design of OTRS is straightforward because all of the additional applications needed are built into one seamless executable. Table 1 shows a table comparing the minimum hardware required by OTRS and what was used.

Table 2: Minimum Required Hardware

Minimum Hardware Requirements	**Hardware Used**
2 GHz Intel® Core 2 Duo (or comparable)	3.3GHz XEON Quad Core Processors
2 GB RAM	4 GB RAM
160 GB Hard Drives	(2) 500 GB Hard Drives
Server Requirements	**Server Used**
Apache 1.3 / 2.x	Apache 2.0.63
Microsoft Internet Information Server (IIS)	Microsoft Internet Information Server (IIS)
Web Browser	**Web browser Used**
All popular XHTML Browsers (Mozilla Firefox, Internet Explorer, Opera etc.)	Internet Explorer 6.0
Database	**Database Used**

MS SQL Server	MS SQL Server
Perl	**Perl Used**
Minimum Perl 5.8	Minimum Perl 5.10

Configuring Fault Tolerance

Microsoft Windows Server 2003 supports Redundant Array of Independent Disks (RAID) 0, 1, and 5 configurations. However, RAID 1, also known as mirroring, was chosen for this project. RAID 1 works by simultaneously copying the same data across two or more separate hard drives. If the primary hard drive fails, the secondary hard drive can step in without a loss in data. A simple visual of RAID 1 is shown in Figure 2. Raid 1 is considered an appropriate hard drive configuration for a small low-volume departmental file server which describes OTRS. As the software development team grows and more data is entered into the system, RAID 1 will eventually be replaced by a back up server.

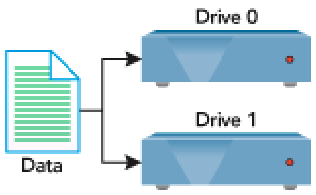


Figure 2: Visual representation of RAID 1

3.2.3 - Build

Appendix B outlines the build procedures performed to put together the OTRS hardware platform and product installation.

3.2.4 - Test & Implementation

After arriving at the Agent Login screen as shown in Figure 3, users are asked to enter their username and password. This screen also allows users to change their default language or retrieve a lost password. A single sign on makes it easy to remember the password and navigate throughout the system without being prompted for another password. The security this page offers is when a person leaves the company or switches departments, their credentials can be removed from the system thus denying access at the next login. The retrieve password feature will not work either because their username will no longer be valid. Agents and Administrators are the only two account types that can create and/or delete usernames.

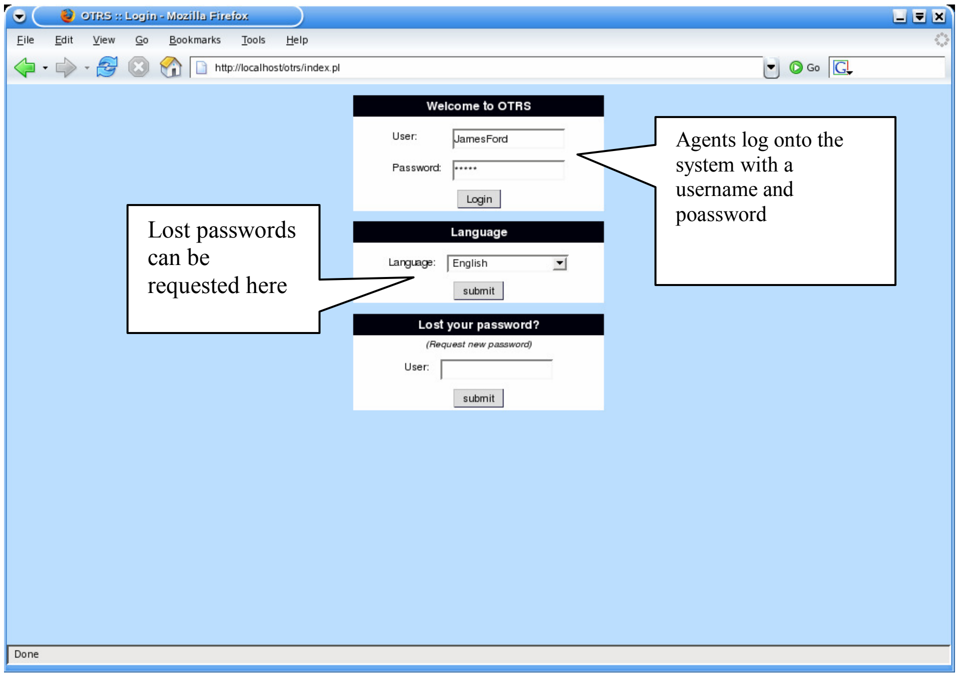
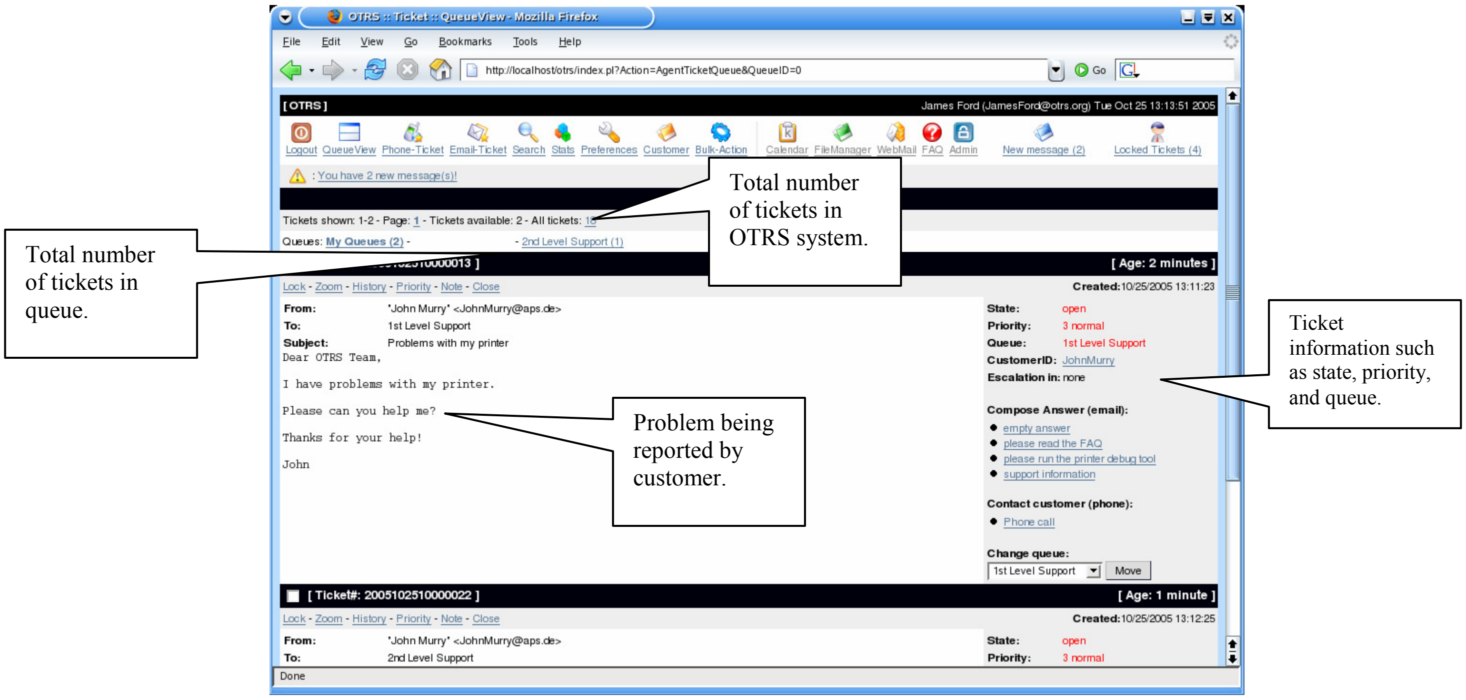

OTRS :: Login - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/otrs/index.pl
Welcome to OTRS
User: JamesFord
Password: *****
Login
Language
Language: English
submit
Lost your password?
(Request new password)
User:
submit
Done
Agents log onto the system with a username and poassword
Lost passwords can be requested here

Figure 3: Agent Login Screen

After clicking on the hyperlink queue users are taken to a screen which shows a list of ticket request that have been assigned to their queue(s).. By default "Junk", "Misc", "Postmaster" and "Raw" queues are created during installation but were modified for the needs of the environment. The “Raw” and “Postmaster” queue was retained because by default they are the repository for all incoming tickets if no queue filters have been created. They are also used to store spam or bogus messages respectively. The “Postmaster” queue was deleted and a “Hardware” queue was created in its place, “Software”, and “Telephone” queues were also created to match the three most problematic areas that occur in the software engineering group.


OTRS :: Ticket :: QueueView - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/otrs/index.pl?Action=AgentTicketQueue&QueueID=0
Go
Logout QueueView Phone-Ticket Email-Ticket Search Stats Preferences Customer Bulk-Action Calendar FileManager WebMail FAQ Admin New message (2) Locked Tickets (4)
Lock - Zoom - History - Priority - Note - Close
Move
Done
Total number of tickets in queue.
Total number of tickets in OTRS system.
Problem being reported by customer.
Ticket information such as state, priority, and queue.

[OTRS] James Ford (JamesFord@otrs.org) Tue Oct 25 13:13:51 2005

: You have 2 new message(s)!

Tickets shown: 1-2 - Page: 1 - Tickets available: 2 - All tickets: 18

Queues: My Queues (2) - - 2nd Level Support (1)

[Age: 2 minutes]

Created: 10/25/2005 13:11:23

From: 'John Murry' <JohnMurry@aps.de>
To: 1st Level Support
Subject: Problems with my printer

```
Dear OTRS Team,

I have problems with my printer.

Please can you help me?

Thanks for your help!

John
```

State: open
Priority: 3 normal
Queue: 1st Level Support
CustomerID: JohnMurry
Escalation in: none

Compose Answer (email):
- empty answer
- please read the FAQ
- please run the printer debug tool
- support information

Contact customer (phone):
- Phone call

Change queue: 1st Level Support

[Ticket#: 2005102510000022] [Age: 1 minute]

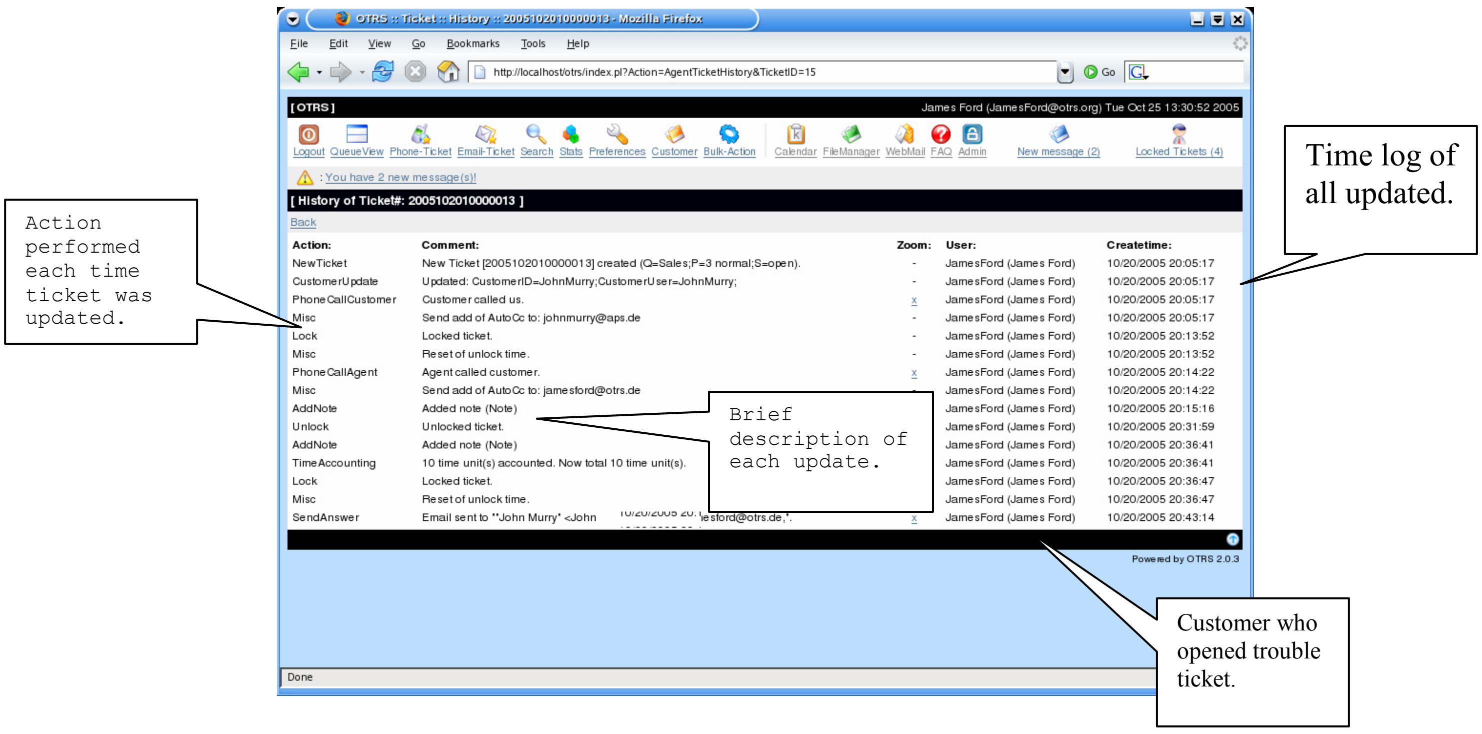
Created: 10/25/2005 13:12:25

From: 'John Murry' <JohnMurry@aps.de>
To: 2nd Level Support

State: open
Priority: 3 normal

Figure 4: Queue View

The ticket history view as shown in Figure 5 is divided into five columns: Action, Comment, User, and Createtime. The action column shows what type of action was performed each time the ticket was updated. Comments column provides a brief description of each update from when the ticket was first created to when the ticket was resolved. The User column shows who reported the problem and Createtime shows the time the change was made. Working in a sensitive environment means the organization sometimes has to submit to random audits. The ticket history view help to access workload and productivity by showing a detailed account of the volume of work and the rate in which tickets were resolved and closed.


OTRS :: Ticket :: History :: 2005102010000013 - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/otrs/index.pl?Action=AgentTicketHistory&TicketID=15
Go
[OTRS]
James Ford (JamesFord@otrs.org) Tue Oct 25 13:30:52 2005
Logout QueueView Phone-Ticket Email-Ticket Search Stats Preferences Customer Bulk-Action Calendar FileManager WebMail FAQ Admin New message (2) Locked Tickets (4)
: You have 2 new message(s)!
Back
Powered by OTRS 2.0.3
Done
Action performed each time ticket was updated.
Time log of all updated.
Brief description of each update.
Customer who opened trouble ticket.

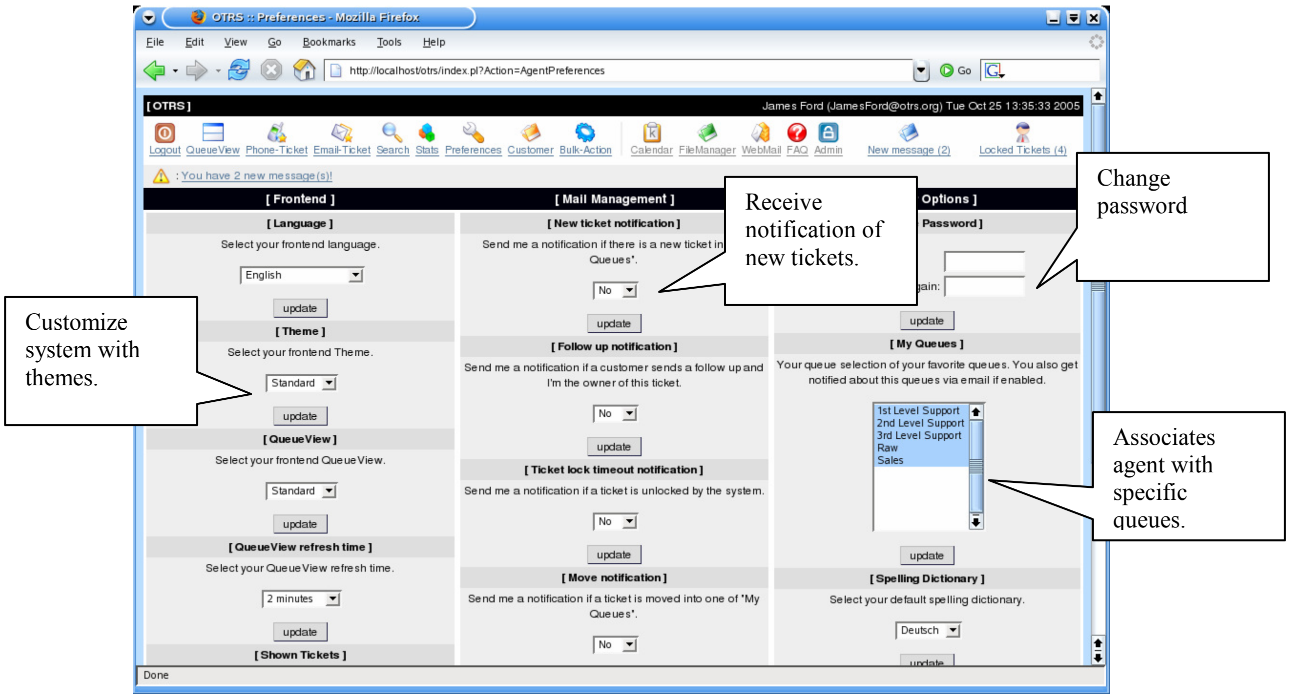
[History of Ticket#: 2005102010000013]

Action:	Comment:	Zoom:	User:	Createtime:
NewTicket	New Ticket [2005102010000013] created (Q=Sales;P=3 normal;S=open).	-	JamesFord (James Ford)	10/20/2005 20:05:17
CustomerUpdate	Updated: CustomerID=JohnMurry;CustomerUser=JohnMurry;	-	JamesFord (James Ford)	10/20/2005 20:05:17
PhoneCallCustomer	Customer called us.	x	JamesFord (James Ford)	10/20/2005 20:05:17
Misc	Send add of AutoCc to: johnmurry@aps.de	-	JamesFord (James Ford)	10/20/2005 20:05:17
Lock	Locked ticket.	-	JamesFord (James Ford)	10/20/2005 20:13:52
Misc	Reset of unlock time.	-	JamesFord (James Ford)	10/20/2005 20:13:52
PhoneCallAgent	Agent called customer.	x	JamesFord (James Ford)	10/20/2005 20:14:22
Misc	Send add of AutoCc to: jamesford@otrs.de	-	JamesFord (James Ford)	10/20/2005 20:14:22
AddNote	Added note (Note)	[illegible]	JamesFord (James Ford)	10/20/2005 20:15:16
Unlock	Unlocked ticket.	[illegible]	JamesFord (James Ford)	10/20/2005 20:31:59
AddNote	Added note (Note)	[illegible]	JamesFord (James Ford)	10/20/2005 20:36:41
TimeAccounting	10 time unit(s) accounted. Now total 10 time unit(s).	[illegible]	JamesFord (James Ford)	10/20/2005 20:36:41
Lock	Locked ticket.	[illegible]	JamesFord (James Ford)	10/20/2005 20:36:47
Misc	Reset of unlock time.	[illegible]	JamesFord (James Ford)	10/20/2005 20:36:47
SendAnswer	Email sent to "John Murry" <John [illegible] esford@otrs.de,'.	x	JamesFord (James Ford)	10/20/2005 20:43:14

Figure 5: Ticket History View

Since the system engineering group (agent) account holders will spend the most of their time using the system, they are provided with many tools for a more customized look and feel. After an agent account holder logs into the system they can select Agent Preferences to access the Agent Preferences View. Figure 6 shows where many application changes can be made to suit individual taste. To start things off, each agent should elect only to receive ticket requests from the queues in which they belong. This prevents having to sort through the general ticket queue to find a particular ticket related to a software service. Instead, a ticket called Software was created and all related ticket requests were automatically routed to that queue. There is also an option to change the ticket queue update intervals, ticket lock timeout notifications, follow-up notifications, and even custom themes.

.


OTRS :: Preferences - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/otrs/index.pl?Action=AgentPreferences
Go
[OTRS]
James Ford (JamesFord@otrs.org) Tue Oct 25 13:35:33 2005
Logout QueueView Phone-Ticket Email-Ticket Search Stats Preferences Customer Bulk-Action Calendar FileManager WebMail FAQ Admin New message (2) Locked Tickets (4)
: You have 2 new message(s)!
Done
Customize system with themes.
Receive notification of new tickets.
Change password
Associates agent with specific queues.

[Frontend]

[Language]
Select your frontend language.
English
update

[Theme]
Select your frontend Theme.
Standard
update

[QueueView]
Select your frontend QueueView.
Standard
update

[QueueView refresh time]
Select your QueueView refresh time.
2 minutes
update

[Shown Tickets]

[Mail Management]

[New ticket notification]
Send me a notification if there is a new ticket in 'My Queues'.
No
update

[Follow up notification]
Send me a notification if a customer sends a follow up and I'm the owner of this ticket.
No
update

[Ticket lock timeout notification]
Send me a notification if a ticket is unlocked by the system.
No
update

[Move notification]
Send me a notification if a ticket is moved into one of 'My Queues'.
No

Options]

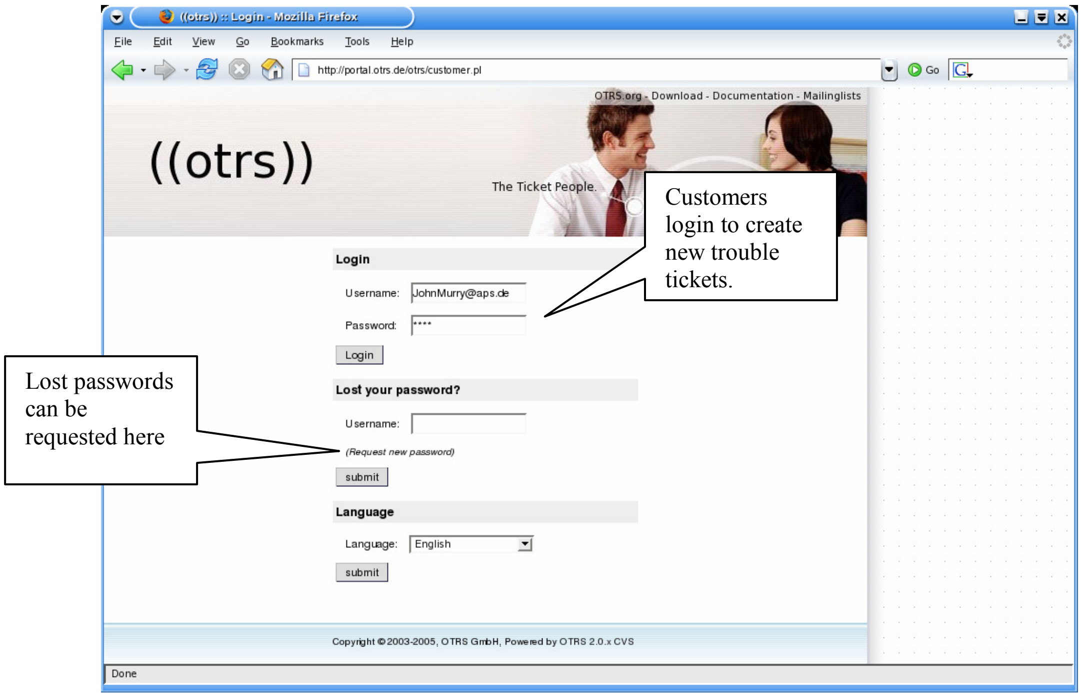
Password]
gain:
update

[My Queues]
Your queue selection of your favorite queues. You also get notified about this queues via email if enabled.
1st Level Support
2nd Level Support
3rd Level Support
Raw
Sales
update

[Spelling Dictionary]
Select your default spelling dictionary.
Deutsch
update

Figure 6: Agent Preferences Screen

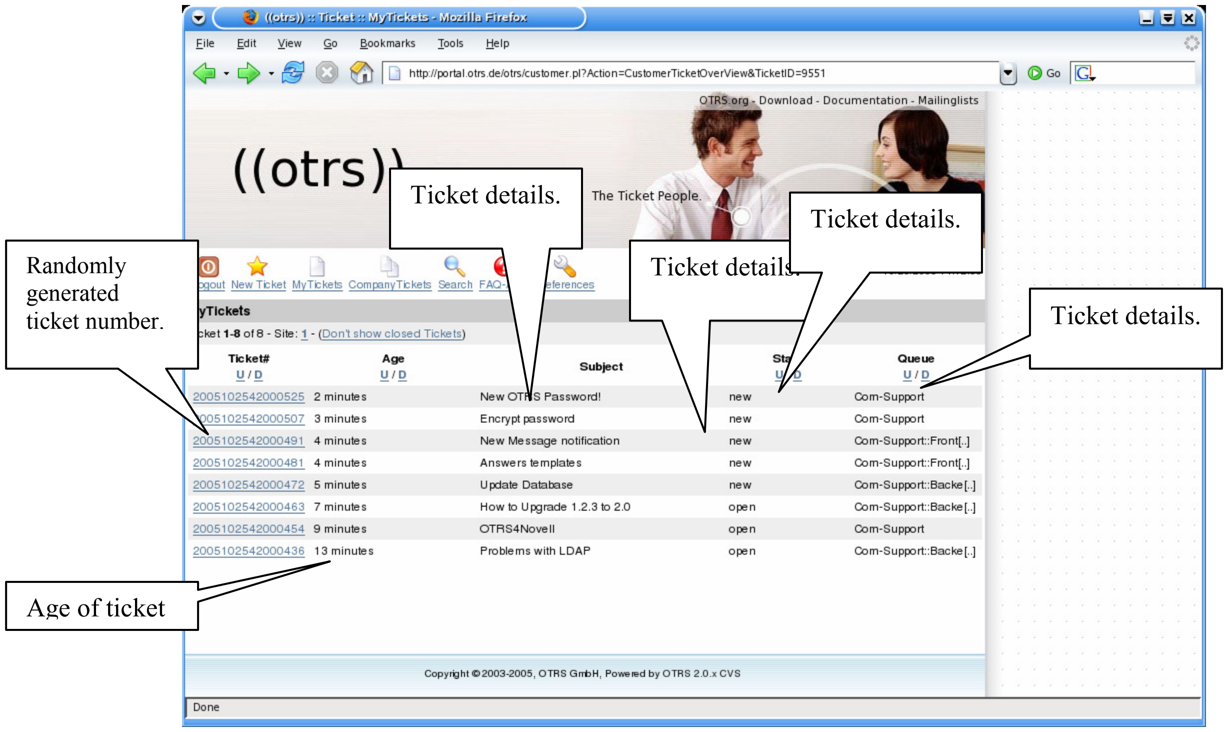
After arriving at the Customer Login screen as shown in Figure 7 users are asked to enter their username and password. This screen also allows users to change their default language or retrieve a lost password. As requested by the software development team, a single sign on makes it easy to remember the password and navigate throughout the system without being prompted to re-enter credentials. From a security standpoint, whenever a person leaves the company or switches departments, their credentials can be easily removed from the system thus denying access at the next login attempt. The retrieve password feature will not longer work because their username will not be found in the system. Agents and Administrators are the only two account types that can create and/or delete usernames.


((otrs)) :: Login - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://portal.otrs.de/otrs/customer.pl
Go
OTRS.org - Download - Documentation - Mailinglists
((otrs))
The Ticket People.
Customers login to create new trouble tickets.
Login
Login
Lost your password?
Lost passwords can be requested here
(Request new password)
submit
Language
submit
Copyright © 2003-2005, OTRS GmbH, Powered by OTRS 2.0.x CVS
Done

Username: JohnMurry@aps.de
Password: ****
Username:
Language: English

Figure 7: Customer Login Screen

After a customer logs into the system they can access MyTickets to view a list of all trouble tickets opened under their user account. Figure 8 shows the MyTicket view which is a simple interface that offers a quick glance of pertinent information for each ticket. MyTickets view shows the trouble ticket number which is randomly assigned whenever a new ticket is created, the age of the ticket which shows how long a trouble ticket has been open, a brief subject which state the problem being reported, the state of the trouble ticket and the responsible queue. For a cleaner view, a user can elect not to show closed trouble tickets.


((otrs)) :: Ticket :: MyTickets - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://portal.otrs.de/otrs/customer.pl?Action=CustomerTicketOverView&TicketID=9551
Go
OTRS.org - Download - Documentation - Mailinglists
((otrs))
The Ticket People.
Ticket details.
Ticket details.
Ticket details.
Ticket details.
Randomly generated ticket number.
Age of ticket
ogout New Ticket MyTickets CompanyTickets Search FAQ- eferences
yTickets
Done

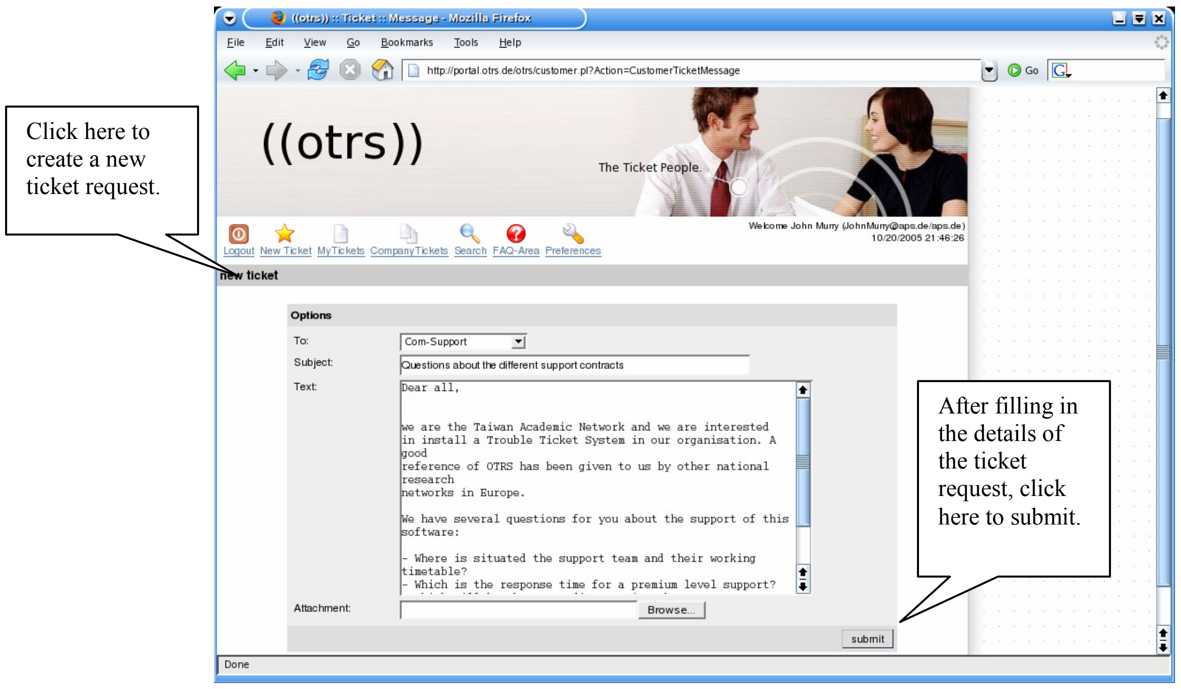
cket **1-8** of 8 - Site: 1 - (Don't show closed Tickets)

Ticket# U / D	Age U / D	Subject	Sta U / D	Queue U / D
2005102542000525	2 minutes	New OTRS Password!	new	Com-Support
005102542000507	3 minutes	Encrypt password	new	Com-Support
2005102542000491	4 minutes	New Message notification	new	Com-Support::Front[..]
2005102542000481	4 minutes	Answers templates	new	Com-Support::Front[..]
2005102542000472	5 minutes	Update Database	new	Com-Support::Backe[..]
2005102542000463	7 minutes	How to Upgrade 1.2.3 to 2.0	open	Com-Support::Backe[..]
2005102542000454	9 minutes	OTRS4Novell	open	Com-Support
2005102542000436	13 minutes	Problems with LDAP	open	Com-Support::Backe[..]

Copyright © 2003-2005, OTRS GmbH, Powered by OTRS 2.0.x CVS

Figure 8: Customer MyTickets View

This is a featured that the software development team really appreciated. As one person stated when asked for feedback about the ease of creating a ticket, “It shows I was listening”. Creating a trouble ticket in OTRS couldn’t be simpler. Once logged into the system, a customer account holder can create a new ticket request by simply clicking the gold star icon at the top of the screen that says “New Ticket”. A “new ticket” window will open providing space to input ticket details. There are only three spaces that need to be updated: The recipient of the ticket or ticket queue, the subject of the ticket, and the problem being reported. There is also an option to include a file attachment which may be a screenshot of an error or a document that further supports the trouble ticket claim. See Figure 9 for a picture of the customer NewTicket screen.


Click here to create a new ticket request.
After filling in the details of the ticket request, click here to submit.
((otrs)) :: Ticket :: Message - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://portal.otrs.de/otrs/customer.pl?Action=CustomerTicketMessage
((otrs))
The Ticket People.
Logout New Ticket MyTickets CompanyTickets Search FAQ-Area Preferences
Browse...
submit
Done

Welcome John Murry (JohnMurry@aps.de/aps.de)
10/20/2005 21:46:26

new ticket

Options

To: Com-Support

Subject: Questions about the different support contracts

Text:

```
Dear all,


we are the Taiwan Academic Network and we are interested
in install a Trouble Ticket System in our organisation. A
good
reference of OTRS has been given to us by other national
research
networks in Europe.

We have several questions for you about the support of this
software:

- Where is situated the support team and their working
timetable?
- Which is the response time for a premium level support?
```

Attachment:

Figure 9: Customer NewTicket

A system administrator account was created for all three system engineers but for best security practices it could only be used for administrative functions. Therefore, the agent account is used for most everyday functions. Administrator access in OTRS is the same as Administrator access in a windows environment. Full access is given to maintain the integrity, reliability and overall health of the system. Anyone with an administrator account have all rights and privileges to make system wide changes therefore it was emphasized that the account be used with caution and any changes made would be approved by the group or at the very least documented. Several administrative tasks that were performed using an administrator account were the creation of user accounts along with groups and queues. SMTP/POP, system logging, and auto responders were also created with administrative privileges. Figure 10 shows the Admin Overview Screen.

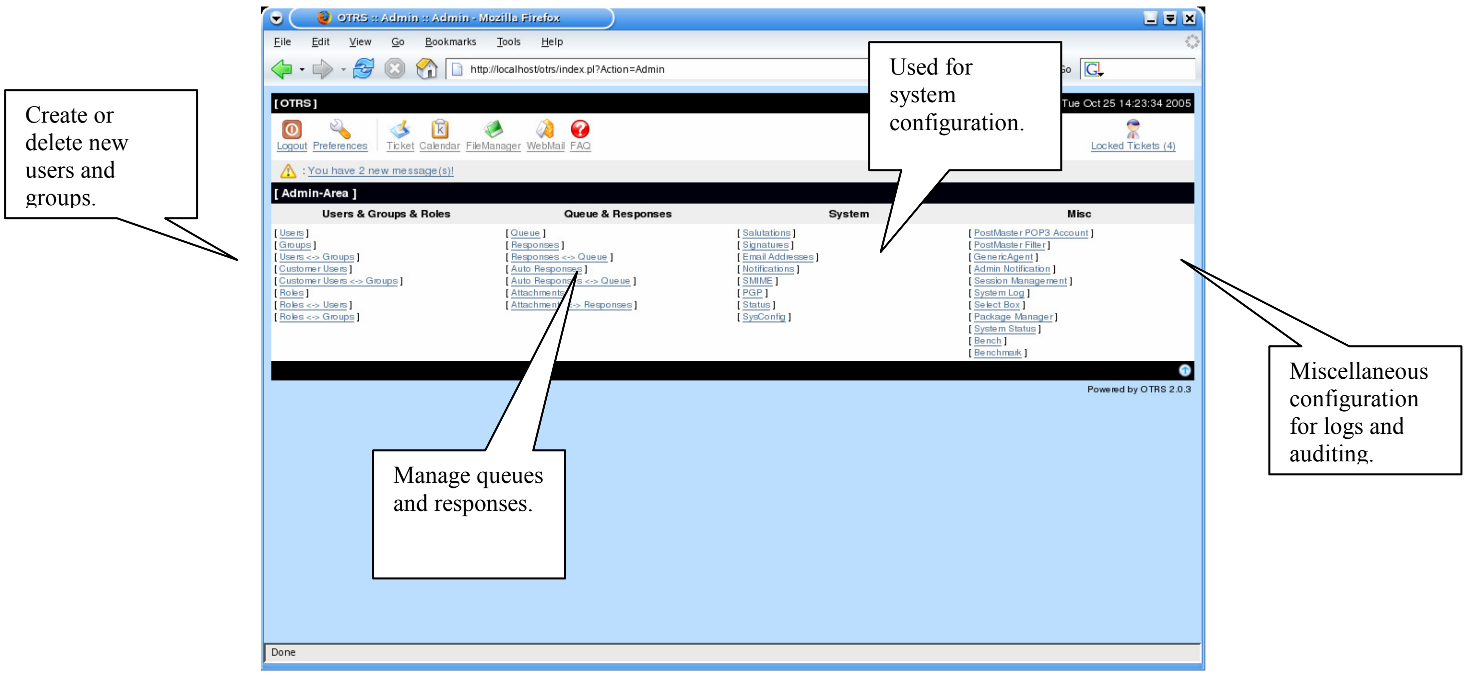

Create or delete new users and groups.
Used for system configuration.
Manage queues and responses.
Miscellaneous configuration for logs and auditing.
OTRS :: Admin :: Admin - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/otrs/index.pl?Action=Admin
[OTRS]
Tue Oct 25 14:23:34 2005
Logout Preferences Ticket Calendar FileManager WebMail FAQ
Locked Tickets (4)
: You have 2 new message(s)!
[Admin-Area]
Users & Groups & Roles
[Users] [Groups] [Users <-> Groups] [Customer Users] [Customer Users <-> Groups] [Roles] [Roles <-> Users] [Roles <-> Groups]
Queue & Responses
[Queue] [Responses] [Responses <-> Queue] [Auto Responses] [Auto Responses <-> Queue] [Attachments] [Attachments <-> Responses]
System
[Salutations] [Signatures] [Email Addresses] [Notifications] [SMIME] [PGP] [Status] [SysConfig]
Misc
[PostMaster POP3 Account] [PostMaster Filter] [GenericAgent] [Admin Notification] [Session Management] [System Log] [Select Box] [Package Manager] [System Status] [Bench] [Benchmark]
Powered by OTRS 2.0.3
Done

Figure 10: Admin Overview Screen

Chapter 4 – Project Analysis and Results

4.1- Did Project Meet Goal?

The goal of the project was to implement an open source trouble ticketing system application that would offer a standardize process to open, track and resolve engineering issues brought forth within the software development group. The OTRS application required minimum modifications therefore it was implemented without any major issues. OTRS not only met the project goals but it exceeded them as well.

As a result of implementing the open source ticketing system application, both engineering groups were able to communicate more effectively and work more efficiently towards their goal of developing high quality software applications. In addition, it became easier to determine the more important tasks and allocate appropriate resources. OTRS made prioritization an easy task. This application will continue to have utility because of its ease of modification for future usage.

4.2- What Went Right and What Went Wrong

Fortunately, most everything went right during the project which may have been the result of not having a baseline of things that could go wrong. Despite the popularity of Linux, it was determined that the OTRS application would be installed on a Dell server running Windows Server 2003 since Linux may poses a learning curve for system engineering group who has the responsibility to maintain the system. The desire to keep this application as simple as possible was paramount throughout build and implementation.

The first major milestone was accomplished after building the hardware needed to host the application and adding and testing Windows Server 2003 without any errors. Because the OTRS application is open source, the compatibility problems between the source code and Windows Server 2003 and/or the hardware were expected but luckily none were found. Both apps seemed to coexist smoothly. The entire design and implementation of the OTRS application was flawless.

4.3- Findings and Analysis and Results

At the start of the project there was limited working knowledge of how a trouble ticket system application is supposed to work but now a more in depth knowledge has been attained. OTRS has all the bells and whistles along with ease of integration to be deployed in small to large enterprise environments. With limited testing the front end web application and back end database applications seems to work flawlessly together. Comments were solicited from some of the software developers who had transitioned over to the new application and considering the limited amount of time spent in the research and implementation phase, all feedback provided was positive. Redundancy will be implemented in future releases.

Chapter 5 – Lessons Learned and Next Evolution of the Project

5.1- Lessons Learned From Project?

This project took the conceptual life cycle models and made them more applicable in a real world situation. Having very little time to complete the project initially was the biggest hurdle to overcome however working with no budget while under pressure was a glimpse of what managers deal with and sometimes on a daily basis.

Many valuable lessons were learned throughout this project but the two most significant lessons learned are the importance of time management and the need to choose hardware carefully. These lessons had an overarching affect on the project as a whole. This project also highlighted the importance of time management when operating on a strict deadline. Because not much time was given to implement this application, a very thorough job of planning and documentation could not be achieved. Documenting the process after the application was implemented was not the most effective approach when striving for accuracy.

This project also emphasized the importance of proper research and analysis. An unsatisfactory job was done while selecting the hardware resources that were used to build this application. If there was no time constraint a more powerful and robust server with larger hard drives would have been selected to accommodate future growth and expansion. Due to these time constraints a random server was selected using hard drives that were available. This seemed like the correct thing to do at the time but as the team grow so will the need for faster high availability resources.

Although the hardware certainly met the minimum specifications of the application, an opportunity to select the best of what was available would have been a much better option. If in the future the application is too resource intensive for the server, the application will have to be migrated over to another server which will result in downtime.

5.2- What Could Have Been Done Differently?

There are a few things that could have been done differently. First, based on the lessons learned regarding time management, there would be a more realistic timeline in order to meet the goals in the most accurate and expeditious manner possible. Second, based on the lesson learned with respect to selection of hardware, there would be the opportunity to have a thorough consideration into future concerns prior to choosing which hardware to use. Third, a proof of concept could have been developed before acquiring and building the hardware needed to host the application. In doing so the performance of the application could have been accessed in a live environment and the amount of hardware resources required would have become more apparent.

5.3- Improvements / Evolution of the project

Currently there are no problems or limitations with OTRS version 2.2.1 thus eliminating the need to immediately upgrade to the newer version 2.2.2 which was released on August 6, 2007. However, as the needs of the project and team grow, there may be a need to consider the newer version.

5.4- Conclusion

Overall, this project provided an excellent learning experience with the importance of setting reasonable expectations when working on a deadline at the forefront. The project was started with a familiarity of trouble ticket system applications but a more in-depth knowledge and understanding was gained during the project life cycle. I now understand how trouble ticket system applications work and how they are implemented in a live environment. Most importantly, this project helped me accomplish all my goals which resulted in the team being able to prioritize critical tasks and work more efficiently in a cost effective manner.

.

References

History of the OSI. Open Source Initiative. 2006. Retrieved from http://www.opensource.org/history on 12 March 2008

Muffatto, Moreno Open Source: A Multidisciplinary Approach. Imperial College Press (2006).12-54.

Srinarayan Sharma, Vijayan Sugumaran & Balaji Rajagopalan, "A framework for creating hybrid-open source software communities", Info Systems Journal 12 (2002): 7–25

Michael J. Gallivan, "Striking a Balance Between Trust and Control in a Virtual Organization: A Content Analysis of Open Source Software Case Studies", Info Systems Journal 11 (2001): 277–304

Holmström, Bengt. "Managerial Incentive Problems: A Dynamic Perspective." Review of Economic Studies. 66:1, 1999. pp. 169-82.

Raymond, Eric. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. Cambridge: O'Reilly 1999. 13-20.

Dixon, Rod. (2004). Open Source Software Law. Artech House

Cowan, C. (2003). Software security for open-source systems. *IEEE Security and Privacy*, 1, 38–45.

Lytras, Toolsby Miltiadis D. & Naeve, Ambjorn. (2007) Open Source for Knowledge and Learning Management: Strategies Beyond. IGI Publishing

Deek, Fadi P. & McHugh, James A. M. (2008). Open Source: Technology and Policy. Cambridge University Press

Appendix A

Installing Windows Server 2003

- Insert Windows Server 2003 CD and restart system. At the Windows Server 2003 Enterprise Edition Setup screen **press Enter** to accept Setup Windows Now.
- At the Windows Licensing Agreement screen **press F8** to agree and proceed.
- Select the default drive for the installation which is typically drive "C" and **press Enter** to start installation.
- Using the arrow keys select "Format the partition using the NTFS file system" and **press Enter**.
- The next screen should ask for confirmation to format the C drive. **Press F** to confirm format. The next screen shows the status of the C drive being formatted. This process may take 5-10 minutes depending on the size of the hard drive. After formatting complete, system files will be copied to the Windows installation folder for preparation.
- At Regional Language and Options Screen **click Next.**
- Enter a name and optional organization and **click Next.**
- Enter product key and **click Next.**
- On the Licensing Mode screen **select Per Device or Per User** depending on the licensing agreement and click "Next".
- On Computer Name and Administrator Password screen, **enter a computer name and Administrator password and click Next.**

- On Modern Dialing Information screen **enter appropriate area code and click Next.**
- On Date and Time Settings screen **enter date, time, time zone, and click Next.**
- On the Network Settings screen **accept the default Typical Settings and click Next.**
- On Workgroup or Computer Domain screen **accept default (WORKGROUP) and click Next.**

Installation will continue……

- After installation, the systems will restart to the login screen. Use the previously created login and password.

After installation is complete run Windows update to apply all Service Packs and Patches

Appendix B

Configuring Fault Tolerance

Before configuring an array, the disks have to be converted from Basic disc to Dynamic disc. The following step shows the configuration using Microsoft Disk Management.

- **Right click** on My Computer icon and **select Manage.**
- **Click on the Disk Management** icon on the lower left hand side of the screen and notice on the right-hand side should be a graphical display of disks that are currently connected to the computer.
- **Right click** on the first disk representation which should be Disk 0 and **click Convert to Dynamic Disk** and **click Yes** when prompted.

A dialog box will warn that file systems will be dismounted.

- **Click Yes** and then **click OK** to restart the computer.

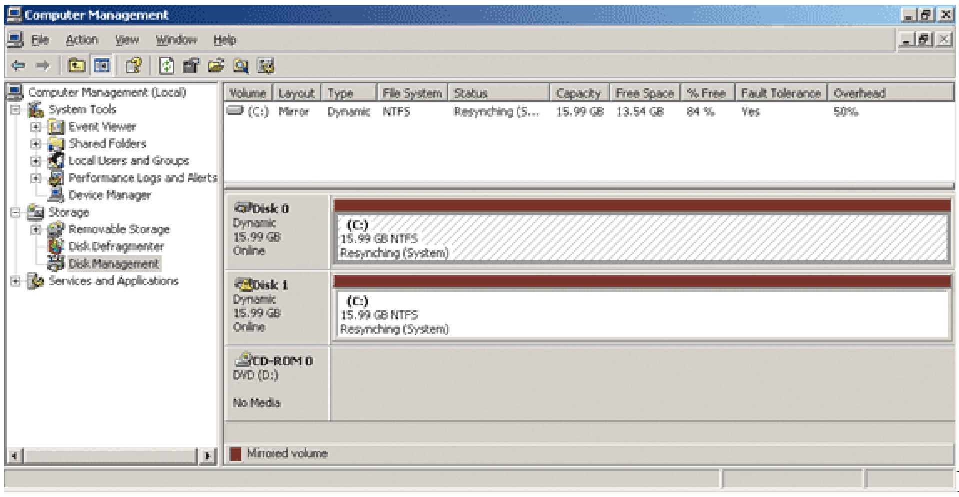
This process requires two restarts. The first restart is needed by Disk Management to complete the conversion of the boot volume device from Basic to Dynamic and the second restart is needed by Plug and Play (PnP) to complete the installation of the new boot volume device. After restarting the system, repeat the process to configure the second disk to convert it to a dynamic disk.

Mirror the Drive

After converting both hard drives to dynamic disc it was time to configure the mirror on the boot volume. The following steps were performed:

- **Right click on the C drive** and then **click Add Mirror.** When prompted to select the location for the mirror **chose Disk 1.**

While the mirror is being built the status will read "Resynching" which means that the two drives are synching. The process isn't complete until the status of the drive changes to "Healthy". After creating the boot mirror I manually simulated a hard drive crash by unplugging the primary hard drive to test whether the system would switch over to the secondary hard drive. I was relieved that it did it!



Volume	Layout	Type	File System	Status	Capacity	Free Space	% Free	Fault Tolerance	Overhead
(C:)	Mirror	Dynamic	NTFS	Resynching (S...	15.99 GB	13.54 GB	84 %	Yes	50%

Disk 0
Dynamic
15.99 GB
Online
(C:)
15.99 GB NTFS
Resynching (System)

Disk 1
Dynamic
15.99 GB
Online
(C:)
15.99 GB NTFS
Resynching (System)

CD-ROM 0
DVD (D:)

No Media

🟥 Mirrored volume

Figure 2 Re-synching Status

Appendix C

Installing OTRS

- Double click on otrs-2.1.4-001-win32.exe
- On the OTRS 2, 1 Setup screen click Next.
- On the Licensing Agreement screen click I Agree.
- Accept the default install location (C:\OTRS) and click Install.
- On the OTRS License screen scroll down and select Accept License.
- On the next screen enter appropriate information OR select defaults and click OK.
- On the next screen click "Finish" to conclude installation.

Configure OTRS Mail

- Log in as Admin Click on "Sysconfig" which should be listed under System.
- On the left side of the screen under Group Selection, make sure Framework is selected and **click Show.**
- On the right side of screen under Results **click Core;:Sendmail.**
- Under Sysconfig make sure "SendmailModule" is checked and "SMTP" is selected in the drop down menu.
- Under the same window make sure "SendmailModule::Host" is checked and **type domain name "mail.domain.com"** in the space provided.
- When finished **click Update.**

Configure OTRS Backup Settings

- Log in as Admin, Click on “Sysconfig” which should be listed under System
- On the left side of the screen under Download Settings, click “Download” to download system config changes
- Save output.

Annotated Bibliography

Optionally, your annotated bibliography can go here.

Glossary

An optional glossary can go here.

Index

An optional index can go here.