U.S. DEPARTMENT OF ENERGY

PNNL-20567

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

NA-42 TI Shared Software Component Library FY2011 Final Report

CK Knudson
FC Rutz
KE Dorow

July 2011

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062;
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@adonis.osti.gov

Available to the public from the National Technical Information Service,
U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161
ph: (800) 553-6847
fax: (703) 605-6900
email: orders@ntis.fedworld.gov
online ordering: http://www.ntis.gov/ordering.htm

This document was printed on recycled paper.
(9/2003)

PNNL-20567

NA-42 TI Shared Software Component Library FY2011 Final Report

CK Knudson
FC Rutz
KE Dorow

July 2011

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99352

1.0 Introduction

The NA-42 TI program initiated an effort in FY2010 to standardize its software development efforts with the long term goal of migrating toward a software management approach that will allow for the sharing and re-use of code developed within the TI program, improve integration, ensure a level of software documentation, and reduce development costs. The Pacific Northwest National Laboratory (PNNL) has been tasked with two activities that support this mission.

PNNL has been tasked with the identification, selection, and implementation of a Shared Software Component Library. The intent of the library is to provide a common repository that is accessible by all authorized NA-42 software development teams. The repository facilitates software reuse through a searchable and easy to use web based interface. As software is submitted to the repository, the component registration process captures meta-data and provides version control for compiled libraries, documentation, and source code. This meta-data is then available for retrieval and review as part of library search results.

In FY2010, PNNL and staff from the Remote Sensing Laboratory (RSL) teamed up to develop a software application with the goal of replacing the aging Aerial Measuring System (AMS). The application under development includes an Advanced Visualization and Integration of Data (AVID) framework and associated AMS modules. Throughout development, PNNL and RSL have utilized a common AMS code repository for collaborative code development. The AMS repository is hosted by PNNL, is restricted to the project development team, is accessed via two different geographic locations and continues to be used. The knowledge gained from the collaboration and hosting of this repository in conjunction with PNNL software development and systems engineering capabilities were used in the selection of a package to be used in the implementation of the software component library on behalf of NA-42 TI.

The second task managed by PNNL is the development and continued maintenance of the *NA-42 TI Software Development Questionnaire*. This questionnaire is intended to help software development teams working under NA-42 TI in documenting their development activities. When sufficiently completed, the questionnaire illustrates that the software development activities recorded incorporate significant aspects of the software engineering lifecycle. The questionnaire template is updated as comments are received from NA-42 and/or its development teams and revised versions distributed to those using the questionnaire. PNNL also maintains a list of questionnaire recipients.

The blank questionnaire template, the AVID and AMS software being developed, and the completed AVID AMS specific questionnaire are being used as the initial content to be established in the TI Component Library.

This report summarizes the approach taken to identify requirements, search for and evaluate technologies, and the approach taken for installation of the software needed to host the component library. Additionally, it defines the process by which users request access for the contribution and retrieval of library content.

2.0 Assumptions

It is expected that software development teams will maintain their own code repositories for use during development and management of their individual applications. The software component library is not intended to become the development environment for all projects but rather a repository where released versions of NA-42 TI program software packages and their associated documentation are stored for reference and available for re-use.

PNNL is responsible for hosting the software component library but does not have the responsibility of ensuring that all NA-42 software development teams comply with NA-42 guidance on the use of or access to the library.

3.0 Requirements

In order to perform an appropriate research and review of existing software packages, it was first necessary to define the functionality and capabilities that were required of the software to meet the needs of the program and the intended end users. Developing a well-defined set of software requirements prior to the initiation of the procurement process benefited the project in a number of ways. First, the requirements provided a consistent set of selection metrics for all staff members involved in the research phase of the procurement eliminating any misdirection and keeping staff members focused. Second, having a pre-defined set of requirements allowed staff members to quickly target prospective applications as well as quickly eliminating inappropriate applications both of which reduced the research and review time. Finally, the requirements provide a template for the testing of the software.

The requirements defined for the TI Shared Software Component Library were developed in three stages. During the first stage, potential categories of users who would be using the software were defined.

3.1. User Stories

The user categories included developers, project managers, system administrators, and testers. The categories of developers and managers were further broken down to include users internal to PNNL and those outside of PNNL. Once the user roles were defined, stage two was to develop a set of user stories to illustrate the desired high level functionality of the software from the perspective of each user category. Table 1 lists the user stories developed for the shared software component library as well as the user categories associated with each story.

Table 1 Component library users and associated functions/stories.

Story #	Function / User Story	TI Mgmt	Internal Developer	External Developer	Internal PM	External PM	System Admin	Tester
1	I want to be able to configure the library so that it is accessible by both internal and external users approved by NA-42						**X**	
2	I want to control access to the library based on roles and responsibilities so I know which users can perform which tasks and provide the necessary accountability						**X**	
3	I need the library capable of operating on publicly available servers so that it is accessible by both internally and externally approved NA-42 users						**X**	
4	I need the library capable of operating on classified networks as well as non-classified networks	**X**			**X**	**X**	**X**	
5	I need the library to be developed by an organization based within the United	**X**			**X**	**X**	**X**	

Story #	Function / User Story	TI Mgmt	Internal Developer	External Developer	Internal PM	External PM	System Admin	Tester
	States							
6	I need sufficient configuration management so I can build any version of the application available within the code library		**X**	**X**	**X**	**X**	**X**	**X**
7	I need sufficient configuration management so I can build new applications using any version of the available software modules within the code library		**X**	**X**				**X**
8	I want a code repository where software applications and modules can be checked in so that approved software developers have them available for review use	**X**	**X**	**X**	**X**	**X**	**X**	**X**
9	I want to develop and store the functions, requirements, and software applications or modules in the code repository so I have traceability between customer/user inputs and the code stored in the library	**X**	**X**	**X**	**X**	**X**	**X**	**X**
10	I want a user interface that provides registration so I can register new components into the library	**X**	**X**	**X**	**X**	**X**	**X**	**X**

Story #	Function / User Story	TI Mgmt	Internal Developer	External Developer	Internal PM	External PM	System Admin	Tester
11	I want a user interface to check in my contributions to the library so I can check in different versions of code modules, source code and documentation	**X**	**X**	**X**	**X**	**X**	**X**	**X**
12	I want a library that can store data sets so I can store data from common data repositories	**X**	**X**	**X**	**X**	**X**	**X**	**X**
13	I want to know the pedigree of the software in the repository so I can choose the best version of the software to meet my needs	**X**	**X**	**X**	**X**	**X**	**X**	**X**
14	I want pedigree information to differentiate between versions of software so my team can best determine which software to use	**X**	**X**	**X**	**X**	**X**	**X**	**X**
15	I want to know the capabilities of the code available for reuse so we can write project proposals that include taking advantage of existing code	**X**	**X**	**X**	**X**	**X**	**X**	**X**
16	I want to search through the library documentation so that I have a single source to search for reusable modules	**X**	**X**	**X**	**X**	**X**	**X**	**X**
17	I want to search through the library metadata so that I have a single source to search for reusable modules	**X**	**X**	**X**	**X**	**X**	**X**	**X**

Story #	Function / User Story	TI Mgmt	Internal Developer	External Developer	Internal PM	External PM	System Admin	Tester
18	I want refined search capabilities so I can create detailed technical descriptions of code modules	**X**	**X**	**X**	**X**	**X**	**X**	**X**
19	I want to check software into the code library that has been developed using a development environment of my choice so that I have a secure location for my software. Acceptance Criteria: Developer checks in code from a framework that is different than the framework of the code library		**X**	**X**				
20	I want to be notified of new bugs against the code my team is responsible for when they are created so I can triage the bug and determine a course of action		**X**	**X**	**X**	**X**	**X**	**X**
21	I want to have access to the software bugs that have been submitted against the code library so I have additional information to determine the quality of the software		**X**	**X**	**X**	**X**	**X**	**X**
22	I want to have access to the software bugs that have been submitted against the code library so I can determine which software will be further developed (or improved upon)	**X**			**X**	**X**	**X**	**X**

Story #	Function / User Story	TI Mgmt	Internal Developer	External Developer	Internal PM	External PM	System Admin	Tester
23	I want to be able to see what software packages I have stored in the repository.	**X**	**X**	**X**	**X**	**X**	**X**	**X**
24	I want to be able to access software information/meta data associated with projects currently in the webPMIS system.	**X**			**X**	**X**	**X**	
25	I want to be able to have the system admin restrict access to certain codes unless the user is authorized by me	**X**			**X**	**X**	**X**	
26	I want to know the system is secure and cannot be penetrated by outside threat	**X**			**X**	**X**	**X**	
27	Xn Roles… I want to ensure that only authorized individuals have access to content within the code library and that authorization will follow a predefined process agreed to by NA-42 and PNNL project management	**X**			**X**	**X**	**X**	
28	I want to be able to place restrictions or caveats on code my team may include in the library	**X**			**X**	**X**	**X**	

3.2. Technical Requirements

The last stage in the development of the software requirements was to break down each of the user stories into technical requirements for the software. These lower level requirements are intended to define the base functionality that the library software would need to meet. Table2 lists the requirements along with the number of the user story they are associated with.

Table 2 Technical requirements and associated user stories.

Story #	Constraints / Requirements
1.1	The system shall provide a way to configure and control two factor authentication for users external to PNNL who have been approved for access by NA-42
1.2	The system shall provide a way to configure and control authentication for users internal to PNNL who have been approved for access by NA-42
2.1	The system must control user access based upon role based authentication
2.2	The system must provide a graphical user's interface that allows an administrator to quickly and easily set and update user accounts
3.1	The system must be capable of running on servers outside of the PNNL firewall
3.2	The system must run on commercially available server equipment
4.1	The software selected for use as the repository must be able to operate on classified and non-classified networks
5.1	The software selected for use as the repository must not be developed or managed by a company outside of the United States
6.1	The software selected for use as the repository must be able to store and maintain different versions of each of the files and projects in the repository
6.2	The software must be able to correlate all stored documentation, release notes, and bug reports associated with each version of software or file stored within the repository
6.3	The software must provide the user the ability to review, check in, and check out any version of a file or project that is stored within the repository
6.4	The system must provide the means to rollback files or projects to earlier versions.
7.1	The software must provide the user the ability to check out individual files, complete projects, and installable executables
8.1	The software must provide the ability to check in new or modified versions of files or projects
8.2	The software must limit the ability to check in new versions of stored files to only the point of contact on record for the stored software
9.1	The software must be able to correlate all stored documentation, release notes, and bug reports associated with each version of software or file stored within the repository
9.2	The system must provide the means for the user to enter and associate tags or keywords with each file or project within the repository
9.3	The system must provide the ability to search files based upon file dependencies
9.4	The system must provide the means for a user to search for files based upon tags, keywords, file type, classification level, language type, and version notes
9.5	The system must provide the ability for a user to access and review all version information for a file or project
10.1	The system must provide a graphical user's interface that allows a user to add files to the repository by browsing to the file or dragging the file icon on top of the interface window

Story #	Constraints / Requirements
10.2	During the file and project add process, the system will provide the user with the means to enter all pedigree information related to the files or projects being added. For each file or project in the repository, this information will include title, type, language developed in, purpose or function of the file, projects or program applications that the files are associated with
10.3	The system must provide a graphical user's interface that allows a user to enter notes and comments for each file or project in the repository
10.4	The system must provide the means for a user to store all files associated with a version of software including code, executable, and documentation files
10.5	The system will record and maintain contact information of the person responsible for each file or project in the repository
11.1	The system must provide a graphical user's interface that allows users the ability to easily check in and checkout files or projects
11.2	During the check in and checkout processes, the system must provide an interface that allows the user to enter notes concerning the check out or check in of the files
11.3	The system must be capable of managing multiple versions of each project or file within the repository
11.4	The system must provide the ability for users to checkout versions of software or files for review and use
11.5	The system must record and maintain checkout and check in information for each version of the files and projects in the repository. This information shall include at a minimum user id, date checked out, date checked in, version checked out, version checked in, and files that were modified during the checkout period
12.1	The system must be able to store any type of file i.e. text, binary, .bas, .com, .dll, .doc, .jpg
13.1	The system must provide the means for the user to enter and associate tags or keywords with each file or project within the repository
13.2	The system must provide the means for a user to enter development or version notes for each file or project stored within the repository
13.3	The system must provide the means for a user to search for files based upon tags, keywords, file type, classification level, language type, and version notes
13.4	The system must provide the ability to search files based upon file dependencies
13.5	The system will provide the means for a user to request and receive contact information for each file or project in the repository
13.6	The system must provide the ability for a user to access and review all version information for a file or project
13.7	The system will provide the means to create and maintain relationships between files or complete projects stored in the repository
13.8	The system will provide the means to search for files or projects by relationships
14	Covered by user story 13
15	Covered by user story 13

Story #	Constraints / Requirements
16	Covered by user story 13
17	Covered by user story 13
18	Covered by user story 13
19.1	The system must be configurable to enable the use of different language compilers
19.2	The system must be able to build any version of a file or project stored in the repository
20.1	The system will automatically notify by email the system administrator and the responsible contact when a bug is submitted to the system for any file or project
20.2	The system will provide the means for a user to enter a new bug message into the system
21.1	The system will provide the means for a developer or tester to enter the current disposition of a bug
22.1	The system will provide the ability for a user to retrieve all bug information associated with a file or project that is stored within the repository
22.2	The system will provide the user with the ability to search for files or projects based upon bug information
23.1	The system must provide the ability for an administrator to automatically generate usage reports including number of files checked out, users who currently have files checked out, number of files or projects in the repository, bug reports, and listing of the names of files and projects currently in the repository
24.1	The system must be accessible for integration with external systems such as webPMIS
26.1	Hardware on which the system will be operating must be capable of being physically locked up to prevent un-authorized access
26.2	The system must not be dependent upon an Internet connection
26.3	The system must provide an approved method of access authentication
28.1	The system must allow for an administrator or the person responsible for the files or projects to limit the access to those files and projects
28.2	The system must allow for the limiting of access based upon role category, user id, client id, and project id

4.0 Technology Evaluation

With the requirements and constraints defined for the TI Shared Software Component Library, a technical evaluation of existing commercial off the shelf (COTS) library software packages was performed. The first step in this evaluation process was the completion of an open source search using the Internet. Through the use of multiple search criteria, a list of thirty-one potential software candidates was developed. This list consisted of both commercial products such as Microsoft's Team Foundation Server software and open source products like TracSVN. Table 3 provides the full list of applications that were reviewed during the selection process.

Table 3 Applications reviewed in library software selection.

TeamForge	Team Foundation Server	PCMDI	Mercurial	Tortoisehg
Git	TortoiseGit	TortoiseSVN	Bazaar	Darcs
Monotone	Perforce	TortoiseCVS	Vesta	JEDI
GNU Arch	BriefCase 3 Toolkit	SourceJammer	SubVersion	PRCS
Aegis	CVS	GNU SCCS	CS-RCS Pro	Atlassian
Dimensions CM	Kiln	StarTeam	Vault	Team Coherence
TracSVN				

The list of candidate applications was further reduced by performing high level reviews against the requirements for each product on the list. This high level review resulted in the list being reduced to three top candidate applications: TeamForge, Team Foundation Server, and TracSVN.

4.1. Down Selection

With the identification of the top three candidate applications complete, Trial versions of TeamForge and Team Foundation Server were installed on test machines for the purpose of testing the basic functionality of each application. This installation of the two software applications also provided the team with the opportunity to evaluate the ease in which the applications were installed and configured. TracSVN is an application that is currently in use on a number of projects and as such the project team was already aware of its use and capabilities so it was not included in the trial version of the review.

Following the trial testing, team members tested each of the applications against each functional requirement and reported on which requirements were not being met. Results of this last review showed that TracSVN only met seventy percent of the requirements and Team Foundation Server only met eighty-five percent of the requirements. The TeamForge from Collabnet was the only application that met 100% of the task requirements. There were two areas in which TeamForge stood out above the other applications, 1) its ease of use, 2) the administration tools that it provides to control access to the system and the objects stored in the repository.

4.2. Final Selection

TeamForge provides a web based GUI that allows a user to select and upload a file. There are free to download desktop applications available through CollabNet which will allow drag and drop capabilities for adding files. These applications include versions for Windows desktop, Visual Studio, and Eclipse. The desktop applications provide more functionality than the web base GUI such as being able to drag and drop

files in order to add them to the repository. All of the desktop applications and the web based interfaces provide the ability for a user to see what tasks, trackers, files, and documents have been submitted by the user as well as the ones assigned to the user. The desktop applications also give the user more methods of viewing and sorting the data and files. In addition, the desktop applications provide graphical user interfaces that allow users to check in and checkout files and code. Using the desktop applications, a user is able to add notes to their check in or check out. The software tracks documents by ID, created by, created date, version, name, size, last edited by, date last edited, and status.

Aside from the usual roles associated with a repository such as administrator, developer, and tester, TeamForge allows the system administrator to define new roles and assign custom access permissions. The software defines user access at two levels, the site administrator level and the project administrator level. At the site administrator level, users are categorized as one of two types, administrator and restricted. Users defined as administrators have unlimited access to all data. Restricted users are limited to accessing only the projects on which they are designated as members. Project managers can add or remove users from accessing the project data as well as defining the user roles associated with the project.

CollabNet, the parent company which develops TeamForge is an established software developer with over ten years of experience and is considered a leader in the development of configuration management and collaboration software. Headquartered in Brisbane, California, CollabNet currently has more than five million users and is in use at more than twenty-five thousand companies world-wide.

Total procurement cost for the TeamForge software was $5,978.87. This price included the annual subscription cost of the software for twenty-five user licenses and CollabNet's Platinum support package. The Platinum support package provides the project with 24 hour access to the CollabNet support staff seven days a week as well as providing initial installation assistance.

5.0 Authentication / Access Control for TICL

5.1. Overview

The intent of this section is to provide a high level overview of the authentication / access control mechanisms that are planned for NA-42 Technical Integration Component Library (TICL). Due to the deficiency in requested vs. actual funding as well as the effects of the continuing resolution (CR) these mechanisms are targeted for implementation in FY12.

5.2. Authentication

A low cost but effective two-factor authentication mechanism is the desired means to handle authentication within TICL. Two-factor authentication implies two independent means of evidence to assert an entity and, in electronic systems, generally takes the form of a user-defined password or Personal Identification Number (PIN) + a value from some sort of physical token. It is our intent to leverage the existing two-factor authentication mechanism already built into WebPMIS as much as possible (given that most of the users of WebPMIS will also be TICL users).

5.3. Access Control

TICL will employ a Role-Based Access Control (RBAC) system to control access to specific content (e.g. access to specific projects). As part of the scope we will determine the required level of granularity needed for content control as well as the level of support within the selected COTS solution for TICL (TeamForge). Based on those requirements, a solution will be designed and implemented. Here again, it is our intent to leverage the RBAC mechanisms already built into WebPMIS as much as possible.

5.4. User Account Management

The authentication and access control mechanisms within TICL will be designed and implemented such that NA-42 management will have full control over who has access into the system and what content they can create / view / edit.

5.5. Audit Capabilities

The TICL authentication and access control mechanisms will provide detailed logging of user activities such that auditing functions can be performed. Examples of logged information would include log-in date / time, content accessed, content created, and functions performed.

6.0 References

2011. CollabNet Overview. www.open.collab.net/about/, Brisbane, California

2011. CollabNet Corporate Overview. www.open.collab.net/media/pdfs/corporate_factsheet.pdf, Brisbane, California

2011. CollabNet Support Options. www.open.collab.net/support/support-programs/, Brisbane, California

2011. CollabNet TeamForge 5.4 Project Administrator Guide. www.Carahsoft.com, Reston, Virginia.

2011. CollabNet TeamForge 5.4 System Administrator Guide. www.Carahsoft.com, Reston, Virginia.

2011. CollabNet TeamForge 5.4 Site Administrator Guide. www.Carahsoft.com, Reston, Virginia.

2011. CollabNet TeamForge 5.4 User Guide. www.Carahsoft.com, Reston, Virginia.

2011. *NA-42 Technical Integration Software Development Questionnaire*. Pacific Northwest National Laboratory. Richland, Washington.

7.0 Appendix 1 Technologies Evaluated

Product	Company	URL
TeamForge	COLLABNET	http://www.collab.net/products/ctf/capabilities.html
Team Foundation Server	Microsoft	http://msdn.microsoft.com/en-us/vstudio/ff637362
PCMDI	CDAT	http://www2-pcmdi.llnl.gov/cdat/support/svn
mercurial		http://mercurial.selenic.com/
tortoisehg	Atlassian	https://bitbucket.org/tortoisehg/stable/wiki/Home
Git		http://git-scm.com/
TortoiseGit		http://code.google.com/p/tortoisegit/
TortoiseSVN		http://tortoisesvn.net/
Bazaar		http://bazaar.canonical.com/en/
darcs		http://darcs.net/
Monotone		http://www.monotone.ca/
Perforce		http://www.perforce.com/perforce/downloads/index.html
TortoiseCVS		http://www.tortoisecvs.org/
Vesta		http://www.vestasys.org/
JEDI		http://jedivcs.sourceforge.net/
GNU arch		http://www.gnu.org/software/gnu-arch/
BriefCase 3 Toolkit		http://www.applied-cs-inc.com/bcintro.html
SourceJammer		http://www.sourcejammer.org/
SubVersion		http://subversion.tigris.org/
PRCS		http://prcs.sourceforge.net/
Aegis		http://aegis.sourceforge.net/
CVS		http://www.nongnu.org/cvs/
GNU SCCS		http://cssc.sourceforge.net/
CS-RCS Pro	ComponentSoftware Inc.	http://www.componentsoftware.com/Products/RCS/index.htm
Atlassian		http://www.atlassian.com/about/
Dimensions CM	SERENA	http://www.serena.com/products/dimensions-cm/index.html
Kiln	Fog Creek	http://www.fogcreek.com/kiln/
StarTeam	Borland	http://www.borland.com/us/products/starteam/index.aspx
Vault	SourceGear	http://www.sourcegear.com/vault/
Team Coherence	QSC	http://www.teamcoherence.com/
Perforce	Perforce	http://www.perforce.com/

8.0 Appendix 2 TeamForge Price Quote

GOVERNMENT- PRICE QUOTATION

COLLABNET GOVERNMENT at CARAHSOFT

12369 SUNRISE VALLEY DRIVE | SUITE D2 | RESTON, VIRGINIA 20191
PHONE (703) 871-8500 | FAX (703) 871-8505 | TOLL FREE (866) 925-OPEN
WWW.CARAHSOFT.COM |OPENSOURCE@CARAHSOFT.COM

TO:	Frederick Rutz Engineer Pacific Northwest National Laboratory 902 Battelle Boulevard P.O. Box 999, MSIN K6-52 Richland, WA 99352	FROM:	Rich Savage Collabnet Government at Carahsoft 12369 Sunrise Valley Drive Suite D2 Reston, Virginia 20191
EMAIL:	frederick.rutz@pnl.gov	EMAIL:	rich.savage@carahsoft.com
PHONE:	(509) 372-4057 FAX: (509) 372-4995	PHONE:	(703) 871-8629 FAX: (703) 871-8505

TERMS:	GSA Schedule No: GS-35F-0131R Term: Nov 19, 2004 - November 9, 2011 FTIN: 52-2189693 Shipping Point: FOB Destination Credit Cards: VISA/MasterCard/AMEX Remit To: Same as Above Payment Terms: Net 30 (On Approved Credit) CAGE CODE: 1P3C5 DUNS No: 088365767 Business Size: Small	QUOTE NO: QUOTE Date: QUOTE EXPIRES: RFQ NO: SHIPPING: TOTAL PRICE: TOTAL QUOTE:	2448879 05/09/2011 06/08/2011 GROUND $5,978.87 $5,978.87

LINE NO.	PART NO.	DESCRIPTION	LIST PRICE	QUOTE PRICE		QTY	EXTENDED PRICE
1	460-105	CollabNet TeamForge - 25 User Native (Annual subscription); Includes Community Support CollabNet - P1025	$4,995.00	$4,783.12	GSA	1	$4,783.12
2	460-291	CollabNet TeamForge Annual Support - Platinum; Per User License CollabNet - U3000-F	$173.00	$47.83	GSA	25	$1,195.75

TOTAL PRICE: **$5,978.87**

TOTAL QUOTE: **$5,978.87**

The Products and Services in this quote are subject to the Terms and Conditions in the link: http://www.carahsoft.com/collabnet/eula/Master%20CollabNet%20Carahsoft%20Hosted%20End%20User%20License%20Agreement.pdf