AD

AD-E403 752

Technical Report ARWSE-CR-15001

REUSE OF THE CLOUD ANALYTICS AND COLLABORATION ENVIRONMENT WITHIN TACTICAL APPLICATIONS (TACAPPS): A FEASIBILITY ANALYSIS

Tiffany A. Reid
EOIR Technologies
Picatinny Arsenal, NJ

Ross D. Arnold
U.S. Army ARDEC
Picatinny Arsenal, NJ

March 2016

U.S. ARMY ARMAMENT RESEARCH, DEVELOPMENT AND ENGINEERING CENTER

Weapons and Software Engineering Center

Picatinny Arsenal, New Jersey

The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

The citation in this report of the names of commercial firms or commercially available products or services does not constitute official endorsement by or approval of the U.S. Government.

Destroy this report when no longer needed by any method that will prevent disclosure of its contents or reconstruction of the document. Do not return to the originator.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-01-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (*From - To*)
March 2016	Final	

4. TITLE AND SUBTITLE

REUSE OF THE CLOUD/ANALYTICS AND COLLABORATION ENVIRONMENT WITHIN TACTICAL APPLICATIONS (TACAPPS): A FEASIBILITY ANALYSIS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS

Tiffany A. Reid, EOIR Technologies
Ross D. Arnold, U.S. Army ARDEC

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

EOIR Technologies
356 Ninth Ave., Suite 102
Picatinny Arsenal, NJ 07806-5000

U.S. Army ARDEC, WSEC
Fire Control Systems &
Technology Directorate
(RDAR-WSF-M)
Picatinny Arsenal, NJ 07806-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army ARDEC, ESIC
Knowledge & Process Management (RDAR-EIK)
Picatinny Arsenal, NJ 07806-5000

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Technical Report ARWSE-CR-15001

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

As part of the Tactical Applications (TacApps) preliminary design effort, the cloud analytics and collaboration environment (CACE) was identified as a potential technology contributor to the TacApps shared workspace. As a follow-on to the preliminary design activities, CACE was analyzed in detail for applicability to the TacApps Program. Methods of analysis included literature review, code inspection, and detailed code analysis. The results of this analysis revealed four specific areas of CACE in which code may be reused for TacApps: notifications (eventing), messaging (including chat), external chat service compatibility, and ozone widget framework inter-widget communication.

15. SUBJECT TERMS

Cloud analytics and collaboration environment (CACE) WebSocket Architecture Messaging Widgets Chat Software Battle command Tactical Applications (TacApps) Command post client (CPC) Command post computing environment (CPCE)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Ross Arnold
U	U	U	SAR	19	19b. TELEPHONE NUMBER (Include area code) (973) 724-8618

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

CONTENTS

FIGURES

ACKNOWLEDGMENTS

The authors would like to thank Timothy Rybarski and Gregory Roehrich from the U.S. Army Armament Research, Development and Engineering Center, Picatinny Arsenal, NJ, for their sponsorship and support, and the Tactical Mission Command Product Management Office for funding the Weapons and Software Engineering Center to undertake this effort.

INTRODUCTION

The cloud analytics and collaboration environment (CACE) is an existing thin client collaboration solution that enables collaboration between Intelligence (G2/S2) and Operations (G3/S3) communities. It is a dashboard-style web application designed to securely host widgets and provide a framework for real-time collaboration between clients, allowing for a plug and play type architecture. Upon an initial review of this application, it appeared to support the core capabilities envisioned by the command post computing environment community for Tactical Applications [TacApps (ref. 1)]. However, as a complete government off-the-shelf (GOTS) solution, CACE is missing two key pieces: disconnected operations and a decentralized server. Despite these drawbacks, CACE still possesses a number of features that are likely to be reusable in TacApps. The reuse of these components would benefit the government by leveraging GOTS software to fulfill several collaboration requirements.

CLOUD ANALYTICS AND COLLABORATION ENVIRONMENT BACKGROUND

The CACE was initially funded by the U.S. Army Intelligence Information Warfare Directorate, as well as other U.S. Army initiatives over the last 6 yr. Some of the large initiatives in which CACE was leveraged include the collaboration and battlespace reasoning and awareness Army technology objective (COBRA ATO), actionable intelligence – technology enabled capability demonstration (AI-TECD), advanced video activity analytics (AVAA), and the cloud analytics for warfighter situational awareness (CLAWS). The CACE fully implements the ozone widget framework (OWF) JavaScript application programming interface (API), allowing OWF widgets to function within CACE without any modifications to their code. In addition to leveraging investments made in the OWF API, CACE offers its own open collaboration API that allows any widget to take advantage of its real-time collaboration via WebSocket communication. The CACE also offers APIs for alerting and file sharing and provides dynamic workflows, cross-widget eventing, and map frameworks. Critical security enhancements were built into the dashboard, removing many of the security flaws present in previous OWF versions. The security enhancements also exist at the data transport layer and server (ref. 2).

CLOUD ANALYTICS AND COLLABORATION ENVIRONMENT ARCHITECTURE

The CACE offers a pluggable architecture in which non-core services can be turned on and off via an administrative service panel. Non-core modules can also be enabled on a per user basis. As part of core functionality, CACE provides eventing, a message center, chat, search, document upload and tagging capability, map data management, and topic forums to facilitate collaboration. The CACE contains three main types of communication: inter-widget, between CACE users, and cross-system data sharing. Communication/collaboration mechanisms include:

- Custom WebSocket implementation
- Representational state transfer
- Java messaging service
- Java application programming interface (API)
- Internet relay chat (IRC)/extensible messaging and presence protocol (XMPP)
- Cursor on target
- Data dissemination service

Figure 1 details the overall CACE architecture represented by a single CACE server, which hosts the CACE application on either a JBoss application server or an Apache Tomcat servlet container instance. The relational database management system can be either PostgreSQL or MySQL. The CACE clients are standard web browser clients. A client is counted as a single browser instance (process) with one tab (browser window). The communication between the CACE clients can occur if they are on the same node as the CACE server or a different node. As shown in figures 1, 2, and 3, the architecture CACE has the ability to host OWF widgets within the workspace and have it appear as a local widget to the end user. Single sign-on is handled within the CACE application, removing the burden from an end user to provide login information to each external widget or chat service (IRC or XMPP).

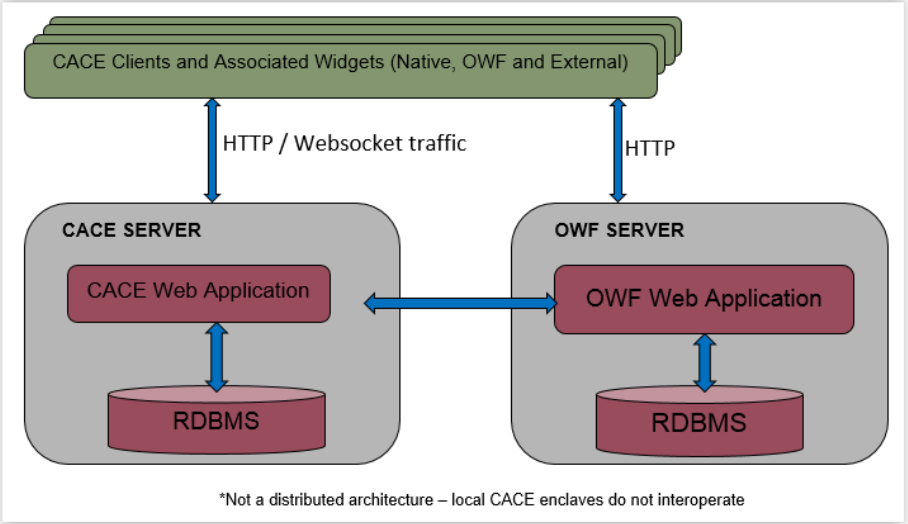

Figure 1
CACE architecture

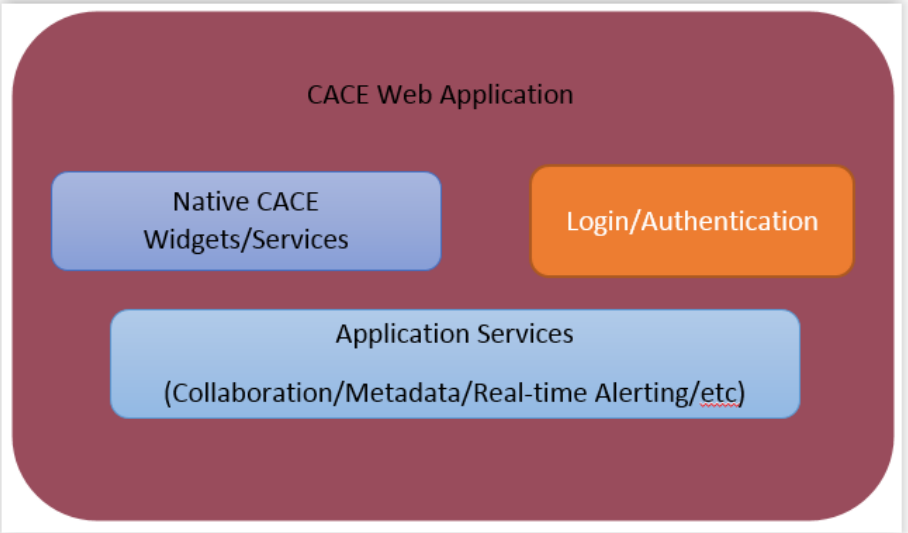

Figure 2
CACE web application components

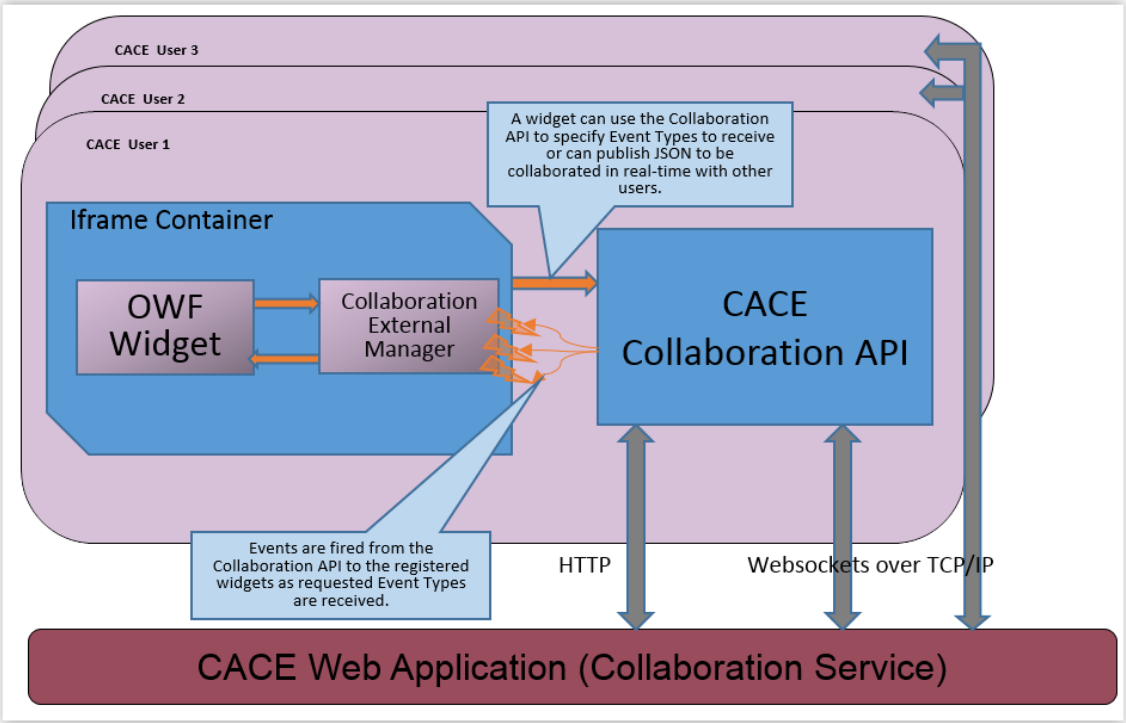

Figure 3
CACE collaboration detail (ref. 2)

The CACE widgets implement the OWF API and, as such, have the ability to respond to OWF events. The OWF widget events are also propagated to multiple CACE users via a client-side publish and subscribe mechanism where events are relayed via JavaScript object notation objects over WebSockets. The CACE's chat component also solely makes use of collaboration over WebSockets for internal chats. Messages received by external chat services are processed by a CACE internal service that mimics the behavior of a typical chat client of the external service. Messages are then relayed to the intended CACE user over WebSockets. In the reverse scenario, messages sent from a CACE client to an external chat source are sent over WebSockets. Once received by the CACE server, these messages are then translated to the messaging format supported by the external service and published for the external recipient who may or may not be a CACE user. This illustrates the versatility of CACE's collaboration beyond registered CACE users. The server-side instance of the CACE application is responsible for the construction of a WebSocket, deconstruction of a WebSocket, and maintenance of internal mappings of open client-to-WebSocket connections. These connections are used as the backbone of client-server communication for relaying events and chat messaging to and from client(s).

SERVICES AND COMPONENTS OF INTEREST

The TacApps has a need to provide a notification/eventing mechanism between its components (ref. 3). Requirements also call out for a chat component that provides both group messages (broadcast capability) and private messages among individual TacApps users (ref. 3). Services and components of interest that are candidates for reuse in TacApps to fulfill these requirements are:

- Notification (eventing)
- Messaging (includes chat)
- External chat service(s) compatibility
- OWF inter-widget communication

LEVERAGING CLOUD ANALYTICS AND COLLABORATION ENVIRONMENT FOR TACTICAL APPLICATIONS

Reuse of CACE components in TacApps will not be a simple plug and play action as the core functionality provided in CACE is tightly coupled. However, reusing CACE components is not an impossible task, and leveraging these pieces will likely result in both schedule and cost-saving benefits.

Chat Component

These CACE object classes and related JavaScript classes are candidates for TacApps reuse.

- com.impulse.messaging.ChatWebSocket – WebSocket class that can process chat messages received by the client, presence updates, and disconnects.
- com.impulse.irc.IRCService - registers as a client for the authenticated user and is able to publish and receive messages to the external IRC service.
- com.impulse.messaging.model.ChatMessage – model of a chat message.
- com.impulse.xmpp.XMPPService – registers as a client for the authenticated user and is able to publish and receive messages to the external XMPP service.

Inter-widget Messaging Component

- User interface (ui)/scripts/iframe/iframebridgeclient.js – supports external widgets that are hosted within an Iframe. Messages or events from those widgets are posted via this class which then coordinates with the browser's cross-window messaging "postMessage" functionality to communicate with the internals of CACE.

- ui/scripts/iframe/iframebridge.js – supports external widgets that are hosted within an Iframe by receiving messages or events that are posted to the browser's cross-window messaging "postMessage" functionality by iframebridgeclient.js. The iframebridge.js can be seen as the gateway into the communication of internals of CACE by outside widgets.

- ui/core/externalmanagers/externalsystemeventdispatcher.js – client side publish and subscribe wrapper for the internal systemeventdispatcher.js, typically invoked by widgets that directly implement the CACE API and also by OWF widgets that are virtually hosted by CACE.

- ui/core/externalmanagers/externaldragdropmanager.js – handles drag drop operation between widgets.

Messaging Component

These CACE object classes and related JavaScript classes are candidates for reuse in TacApps.

- com.impulse.eventing.SystemEventDispatcher – responsible for transaction synchronization and coordination of message dispatching. Accepts system event messages that are of type serializable and sends them to MessagingServlet.

- com.impulse.messaging.AbstractWebSocket – base WebSocket class that is aware of how to broadcast messages and disconnect clients.

- com.impulse.messaging.InboundWebSocket – general WebSocket class that handles ALL events and/or messages not of type serializable that are received from the client.

- com.impulse.messaging.MessagingServlet – maintains a running list of open WebSockets with corresponding user for message distribution. Receives system events (including chat messages) and publishes them to the client.

- ui/core/managers/systemeventdispatcher.js – client side publish and subscribe component. Handles and maintains listeners that require a callback when system events occur.

- ui/scripts/jjms.js – client side WebSocket component responsible for initiating construction and de-construction of WebSockets. Also handles default processing on receipt of messages sent via WebSocket.

Ozone Widget Framework

These CACE JavaScript classes are candidates for reuse in TacApps.

- ui/core/framework/owfsupport.js – handles all OWF widget events and communication.

It is important to note that the OWF functionality makes use of and is dependent upon the inter-widget messaging component mentioned previously (fig. 4).

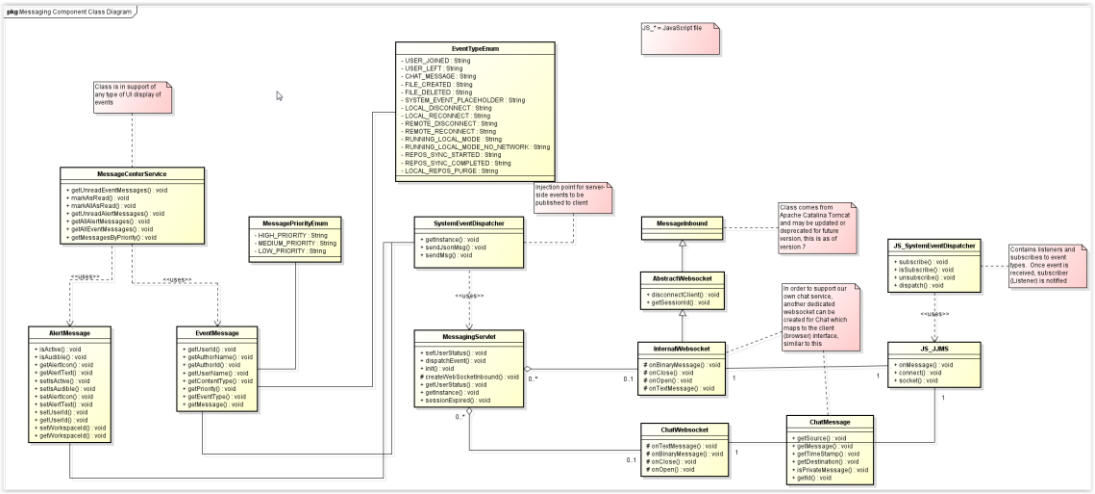

Figure 4
TacApps messaging component class diagram (ref. 4)

TACTICAL APPLICATIONS COMPONENT INTEGRATION

Tactical Applications Messaging Component

The core of CACE message delivery system is supplemented by additional classes for TacApps specific functionality. Supplemental classes of interest are:

- MessagingCenterService – service that provides message and alert management functionality to the TacApps UI component.
- MessagePriorityEnum – assists in prioritizing messages for UI management.
- AlertMessage – user-defined message of interest upon which delivery is triggered on situational changes/updates.
- EventTypeEnum – used to differentiate messages for delivery. Recipients may opt on receipt of specific messages only and tailor consequential actions appropriately.

Tactical Applications Chat Service Component

The class diagrams for TacApps ChatCoordinatorService, internal and external chat managers are shown in figures 5 through 7.

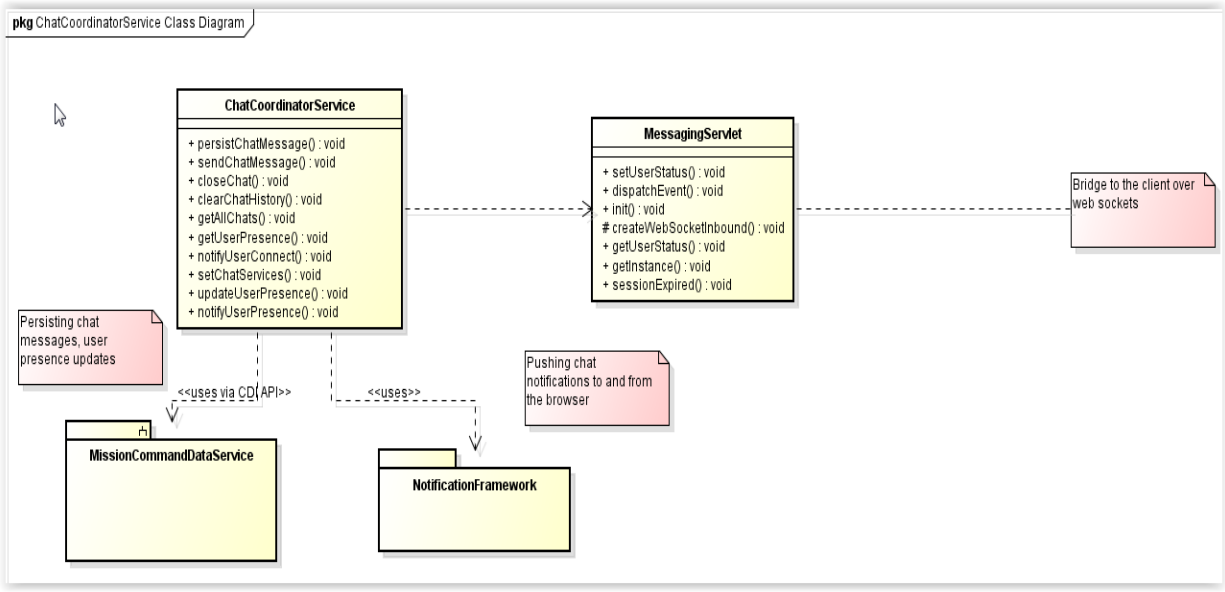

Figure 5
TacApps ChatCoordinatorService class diagram (ref. 4)

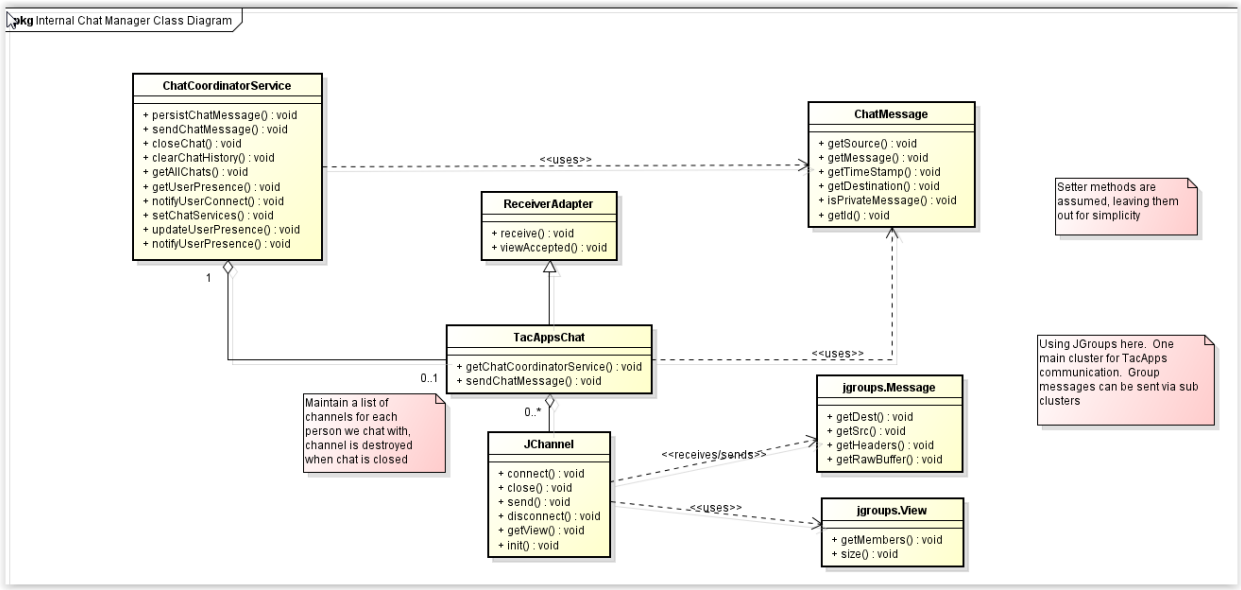

Figure 6
TacApps internal chat manager class diagram (ref. 4)

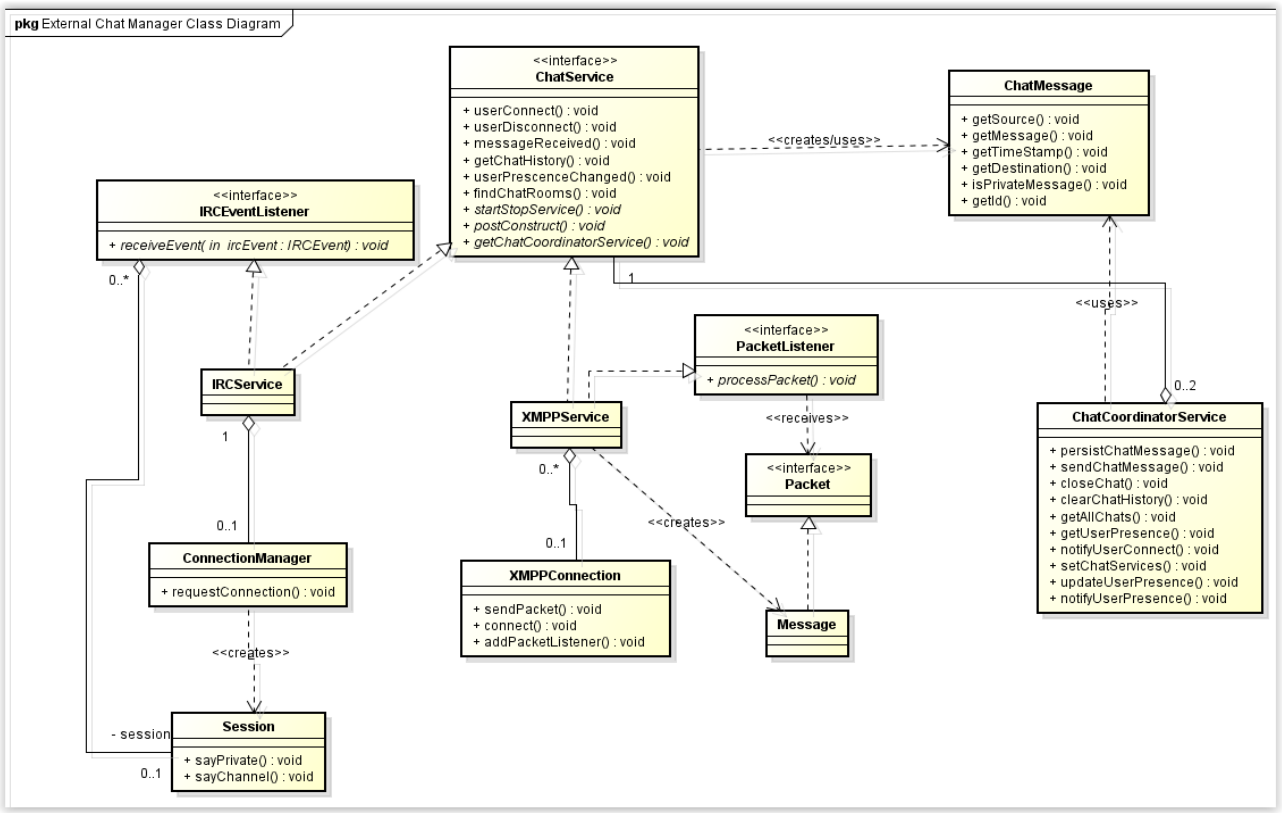

Figure 7
TacApps external chat manager class diagram (ref. 4)

The TacApps chat component borrows external chat service integration from CACE; however, the chat component updates the internal chat mechanism (TacApps-to-TacApps) to work synergistically with the TacApps architecture. This design is necessary because it is not feasible to depend on WebSockets for client to server communication across nodes that already host a client-to-server local solution (as needed for disconnected operations). The standout classes are:

- ChatService – interface of which all external chat services must comply at minimum. TacApps will reuse CACE's XMPPService and IRCService as implementers of the ChatService. The ChatService closely resembles CACE's ChatSink class; however, additional messages are part of this new interface that is pertinent to TacApps. Updates are planned to be made to the implementing classes to comply.

- ChatCoordinatorService – coordinates messages between UI and TacApps-to-TacApps as well as coordinating messages between UI and TacApps-to-External Chat Service.

- TacAppsChat – facilitates instant messaging between TacApps-to-TacApps by using a brokerless service as the messaging backbone to promote auto discovery among TacApps nodes.

ISSUES AND RISKS

Three of the identified risks are:

- **Risk 1 -** JavaScript calls to the server for service functionality within CACE are currently done through a Java library called direct web remoting. This library has been part of the core CACE architecture for quite some time; however, there have not been new developments in this library since September 2014. The most recent version of the library is version 3.0.0-rc3; however, this version is not compatible with CACE as obtained directly from http://directwebremoting.org/dwr/index.html.

- **Risk 2 -** Analysis was performed on Apache Tomcat 7; as of April 2015, CACE officially supports Apache Tomcat 7 and JBoss 7. The CACE has not been officially tested with WildFly (official name for JBoss 8) nor Apache Tomcat 8. The CACE makes updates to the deployment of Apache Tomcat 7 and JBoss 7; however, TacApps team developers were informed by CACE developers that these server updates in support of WebSockets would not be necessary in the future server versions.

- **Risk 3 -** Although OWF widgets can be virtually hosted within CACE, the reverse is not true; CACE widgets cannot be hosted by an OWF server as the unique collaboration of widgets across multiple clients will be lost. This is important to note with TacApps since one of TacApps goals is to interoperate with OWF. It is worth considering whether or not CACE's approach regarding a unidirectional OWF compatibility is what is needed in TacApps. By traversing a bi-directional path of OWF compatibility, TacApps widgets may lose inter-widget communication across multiple clients (which is the desired behavior). Further investigation is needed for the latter approach when hosted on an OWF server.

CONCLUSIONS

The Tactical Applications (TacApps) Program is likely to benefit from reusing the core components and services of cloud analytics and collaboration environment (CACE) related to collaboration of:

- Notification (eventing).

- Messaging (includes chat).

- External chat service(s) compatibility.

- OWF inter-widget communication.

Although the architecture for TacApps differs from CACE in that TacApps data is distributed, it uses a non-centralized server, and has the capability of operating under disconnected intermittent low-bandwidth conditions; the communication requirements between the client-to-server(s) and client-to-client remain the same. By reusing the aforementioned components and services, the TacApps Program will be leveraging technology that has been proven in a program for the U.S. Army. This is not to imply that it would be a simple plug and play solution as the CACE functionality is tightly coupled within the architecture. However, by using the key classes and applying design modifications as outlined in this report (and further detailed in the tactical applications software design document) to comply with TacApps requirements and architecture, the benefits of this reuse are clear.

REFERENCES

1 Arnold, R.D., Lieb, A.J., Samuel, J.M., Burger, M.A. “The Next Generation of Command Post Computing,” Proceedings of the 2015 SPIE Defense Systems & Security Conference: Baltimore, MD, 9456-30, April 2015.

2 Cloud Analytics and Collaboration Environment (CACE) Overview, presented by EOIR Technologies, at the U.S. Army Armament Research, Development and Engineering Center (ARDEC), Picatinny Arsenal, NJ, 2014.

3 U.S. Army Armament Research, Development and Engineering Center (ARDEC), Picatinny Arsenal, NJ, Weapons and Software Engineering Center. Tactical Applications Software Requirements Specification v1.0, 2015.

4 U.S. Army Armament Research, Development and Engineering Center (ARDEC), Picatinny Arsenal, NJ, Weapons and Software Engineering Center, Tactical Applications Software Design Description v1.0, 2015.

DISTRIBUTION LIST

U.S. Army ARDEC
ATTN: RDAR-EIK
RDAR-WSF-M, R. Arnold
RDAR-WSH-H, T. Reid (Contr. EOIR)
Picatinny Arsenal, NJ 07806-5000

Defense Technical Information Center (DTIC)
ATTN: Accessions Division
8725 John J. Kingman Road, Ste. 0944
Fort Belvoir, VA 22060-6218

GIDEP Operations Center
P.O. Box 8000
Corona, CA 91718-8000
gidep@gidep.org

REVIEW AND APPROVAL OF ARDEC TECHNICAL REPORTS

Reuse of Cloud Analytics and Collaboration Environment Within TacApps: A Feasibility Analysis

Title

Date received by LCSD

Tiffany A. Reid
Ross D. Arnold

Author/Project Engineer

Report number (to be assigned by LCSD)

X8618 31

Extension Building

RDAR-WSF-M

Author's/Project Engineers Office (Division, Laboratory, Symbol)

PART 1. **Must be signed before the report can be edited.**

a. The draft copy of this report has been reviewed for technical accuracy and is approved for editing.

b. Use Distribution Statement A__x__, B__, C___, D____, E____, F____ or X____ for the reason checked on the continuation of this form. Reason: _____

1. If Statement A is selected, the report will be released to the National Technical Information Service (NTIS) for sale to the general public. Only unclassified reports whose distribution is not limited or controlled in any way are released to NTIS.

2. If Statement B, C, D, E, F, or X is selected, the report will be released to the Defense Technical Information Center (DTIC) which will limit distribution according to the conditions indicated in the statement.

c. The distribution list for this report has been reviewed for accuracy and completeness.

Patti Alameda 5/14/15
Division Chief (Date)

PART 2. To be signed either when draft report is submitted or after review of reproduction copy.

This report is approved for publication.

Patti Alameda 5/14/15
Division Chief (Date)

Andrew Pskowski 3/30/16
RDAR-CIS (Date)