Improving Situation Awareness with the Android Team Awareness Kit (ATAK)

Kyle Usbeck[a], Matthew Gillen[a], Joseph Loyall[a],
Andrew Gronosky[a], Joshua Sterling[b], Ralph Kohler[c]
Kelly Hanlon[d], Andrew Scally[d], Richard Newkirk[c], and David Canestrare[c]

[a]BBN Technologies, Cambridge, MA, USA
[b]NVESD, Fort Belvoir, VA, USA
[c]Air Force Research Laboratories, Rome, NY, USA
[d]WinTec Arrowmaker, Tampa, FL, USA

ABSTRACT

To make appropriate, timely decisions in the field, Situational Awareness (SA) needs to be conveyed in a decentralized manner to the users at the edge of the network as well as at operations centers. Sharing real-time SA efficiently between command centers and operational troops poses many challenges, including handling heterogeneous and dynamic networks, resource constraints, and varying needs for the collection, dissemination, and display of information, as well as recording that information.

A mapping application that allows teams to share relevant geospatial information efficiently and to communicate effectively with one another and command centers has wide applicability to many vertical markets across the Department of Defense, as well as a wide variety of federal, state local, and non-profit agencies that need to share locations, text, photos, and video.

This paper describes the Android Team Awareness Kit (ATAK), an advanced, distributed tool for commercial-off-the-shelf (COTS) mobile devices such as smartphones and tablets. ATAK provides a variety of useful SA functions for soldiers, law enforcement, homeland defense, and civilian collaborative use; including mapping and navigation, range and bearing, text chat, force tracking, geospatial markup tools, image and file sharing, video playback, site surveys, and many others. This paper describes ATAK, the SA tools that ATAK has built-in, and the ways it is being used by a variety of military, homeland security, and law enforcement users.

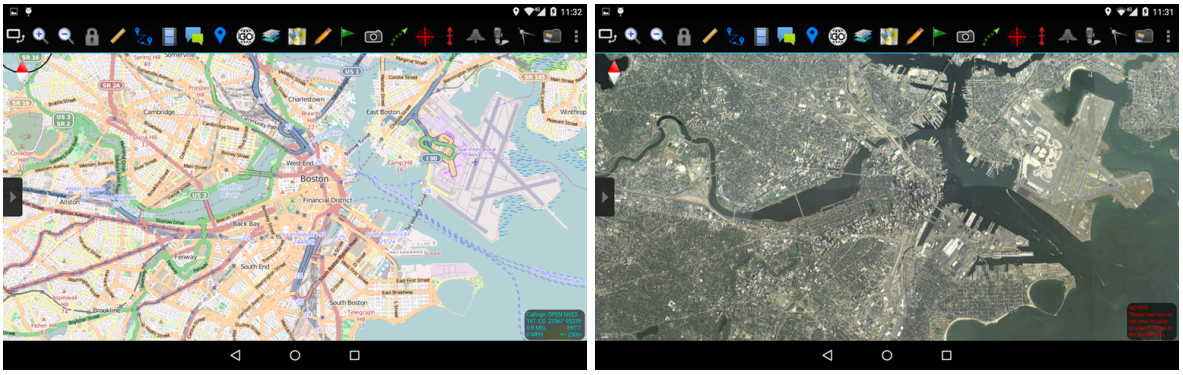
Keywords: situational awareness, Android, mapping

1. INTRODUCTION

Maps and situational awareness are critical to coordinated tasks in many contexts, including kinetic military action, law enforcement, civil protection, disaster relief, and many others. Geospatial data has been the purview of specialists for some time, and uniting up-to-date geospatial data with real-time situational awareness has been difficult. We present the Android Team Awareness Kit (ATAK) as a solution to these problems. ATAK is a Situational Awareness (SA) app that runs on commodity Android-based smartphones and tablets and provides a wide variety of mapping and SA features for users. ATAK has the ability to display a wide variety of map data either stored on the device (i.e., offline maps) or downloaded in real-time from a Web Map Tile Service (WMTS) such as Google Maps or Digital Globe. In addition, ATAK has a wide variety of ways to share SA data, completely separate from the source of the map data. Finally, ATAK has a selection of tools designed to facilitate real-time coordination between team members.

This paper describes the key aspects of ATAK. First we will discuss SA, what it is, why it is important, and the three main components of SA; namely *maps*, *real-time data*, and *mission-specific tools*. We will describe

The work described in this paper was supported in part by the US Air Force Research Laboratories (AFRL).
Further author information: (Send correspondence to KU)
KU: kusbeck@bbn.com, JL: jloyall@bbn.com, MG: mgillen@bbn.com
AG: agronosk@bbn.com, JS: joshua.d.sterling.civ@mail.mil, RK: ralph.kohler@us.af.mil



(a) Vector map (© OpenStreetMap contributors) (b) Imagery map

Figure 1: Example map types loaded in ATAK

how ATAK supports each of those three SA components. Next we will discuss some of the built-in tools that are currently implemented in ATAK. Then we will talk about ongoing work on making ATAK extensible; why extensibility is important and the mechanics of what we are implementing. Finally, we will talk about some of the users of ATAK and how they are using it.

2. SITUATIONAL AWARENESS

Situational awareness is important to the execution of military missions at all levels, from strategic to tactical. Its importance has been highlighted by the earliest writers of military doctrine; Sun Tzu talks extensively about maintaining your own SA while denying SA to your adversary. Concrete definition of SA is tricky though, partly because the specifics of what constitutes "good" SA is context- and mission- dependent.[1] For instance, a diplomatic mission might consider social, economic, and political intelligence as the primary drivers of SA, while a squad conducting a sweep of an area might consider maps with mission-plan overlays along with current and past positions of squad members as the primary drivers of SA.[1] contains a succinct and general definition of SA:

> Continuous extraction of environmental information, integration of this information with previous knowledge to form a coherent mental picture, and the use of that picture in directing further perception and anticipating future events.

ATAK focuses on improving the SA of small units at the tactical edge. SA at the tactical edge means knowing where you are, where the rest of your team is, and having a variety of ways to communicate with your team (and, if feasible with reach-back, to operation centers).

2.1 Maps

Maps, and knowing where you are on them, are often the most important aspect of SA, and this is especially true on the tactical edge. Maps have a wide variety of uses and specializations, as illustrated in Figure 1. Some maps, like the ones produced by OpenStreetMap (an example OpenStreetMap map of Boston is shown in Figure 1a), focus on roads, political boundaries, and landmarks. Other maps focus on aerial or satellite-based imagery as in Figure 1b. Still others focus on elevation changes and tall obstacles (most often needed for flight-planning). Finally, some maps are hybrid combinations of these (e.g. showing road names overlaid on satellite imagery).

For as many different types of maps there are, there almost as many different digital formats for map data. The Army Geospatial Center (AGC) recently did a trade study of Android-based map engines, and identified eight core data formats that the map engines they were evaluating needed to support.[2] These formats are:

DTED	Digital Terrain Elevation Data
GeoTIFF	Geographic Tagged Image File Format
JPEG2000	Joint Photographics Expert Group 2000
NITF	National Imagery Transmission Format
MrSID	Multi-Resolution Seamless Image Database
Vector overlays	Esri Shape Files, Keyhole Markup Language (KML), and others
WMTS	Open Geospatial Consortium's Web Map Tile Service standard; a protocol often used by commercial map providers such as MapQuest or Google Maps used to acquire portions (or tiles) of a map

This variety of format support is necessary because of the variety in map sources and the type of data that they contain. It is worth noting as well that among the data formats specified are three fundamentally different kinds of data: elevation, raster imagery, and vector overlays. All of these are important, but they have different epochs of relevancy. Elevation data rarely changes. Raster data can change as often as overhead flights or satellites pass over an area. The vector data can be relatively static (e.g., roads) or highly mission-specific (e.g., locations of perimeter defense sensors).

ATAK supports all the formats from the AGC study and the following additional formats:

- A long list of Raster Product Format (RPF) subcategories, like Compressed arc-second raster chart/map (ARC) Digitized Raster Graphics (CADRG) and Controlled Image Base (CIB)
- Local Point and Draw files – vector data formats primary used in the Falconview mapping application (https://www.falconview.org/)
- Portable Reference Imagery (PRI) and Precision Fires Imagery (PFI) – variations on the standard National Imagery Transmission Format (NITF) format that can be used for high-precision localization
- Global Positioning System (GPS) Exchange Format (GPX) – vector data format commonly used by commercial GPS devices
- Digital Aeronautical Flight Information File (DAFIF) – raster image format used for flight planning
- Gridded Reference Graphics (GRG) – mission planning data in the form of raster imagery and vector graphic overlays

2.1.1 Projections

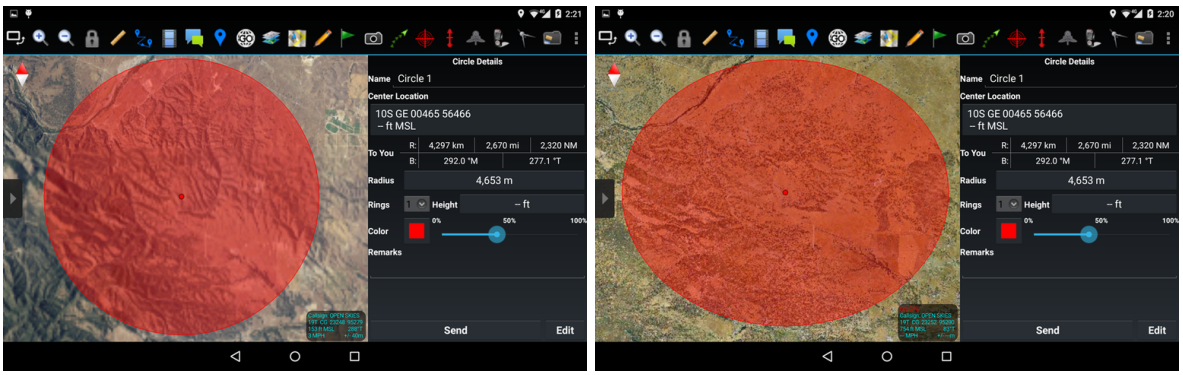
Raster map data from aerial or satellite sources needs to be georeferenced in order to be useful. There are several components to georeferencing, but one aspect is warping the imagery to account for the curvature of the earth. To warp the imagery, a map projection must first be chosen. This warping operation is computationally expensive, since it typically involves complex calculations for each pixel in the raster.

This issue of projections and warping is important for practical reasons. Warping and re-projection is typically done by powerful server machines, and while in theory all map data can be pre-processed, in practice it can be difficult to keep up with all the possible imagery products. It is also not feasible or wise to try and re-project on mobile devices, because the processing power is already very limited. Trying to re-project on a mobile device would take exorbitantly long and make the device unresponsive.

So mobile devices are faced with three issues: raster map data is distributed in a variety of projections, it is not practical to pre-process map data into a single projection, and the raster data cannot be re-projected on the fly because of processing constraints. ATAK solves this conundrum by changing the projection it uses based on the raster map layer currently in use. This helps ATAK stay responsive no matter what kind of map data is in use.

Changing the projection also affects the vector data, but it only affects the visual appearance, not the computational load. Plotting vector data is done by a straightforward translation into OpenGL objects, and drawing those objects on the screen costs the same computationally regardless of the map projection in use.

Some of the effects of changing a map projection are shown in Figures 2a and 2b. These figures show the the same range-circle (which has a constant radius) with map data using different projections. You can see in one figure a round looking circle, while in the second figure the circle looks more like an oval. The change in visual appearance of that constant-radius circle is driven purely by the change in map projection, which in turn was caused by using two map layers that were constructed using separate projections.



(a) Uniform-radius circle appears circular. (b) Uniform-radius circle appears ellipsoidal.

Figure 2: The same uniform-radius circle annotation viewed in different map projections.

While ATAK solves many problems by adjusting the map projection on the fly, there is a theoretical drawback in that only raster data from a single projection can be displayed simultaneously. A potential use case where transparency of one raster layer is used to overlay another raster layer could only work in ATAK if both layers use the same projection.

2.2 Real-time SA

Maps change infrequently relative to specific missions and positions of people and movable objects. We have separated the data that can change frequently into a separate category, namely "real time SA." Things that fall into this category include current positions of people and vehicles, video sources, and various notification and communications mechanisms. This section will discuss each type of real-time SA data in depth.

Blue force tracker (BFT) was a revolutionary system that used GPS and satellite-based communications (SATCOM) to great effect. The idea is that every vehicle and every soldier carries a transmitter that pushes out its position to a variety of upstream systems. The SA picture that BFT provided improves coordination and reduces the possibility of fratricide. ATAK fills a similar role, but it has the advantages that it runs on commodity commercial off the shelf (COTS) hardware and can use a variety of communication substrates (not just SATCOM). We will discuss the various communication substrates ATAK can use in a later section, but given an Internet Protocol (IP) layer, ATAK can share its own position with other ATAK devices and SA displays, and display positions of blue forces that are advertising their position on the network.

Another resource that is common to tactical environments is video. Whether from a Unmanned Aerial Vehicle (UAV) or from Unattended Ground Sensors (UGS), video can contribute significantly to SA, especially when it comes to red-force tracking (the opposition forces do not usually wear tracking devices so that we can plot their positions in real time). ATAK has a built-in video player that can play video from 95% of the video encoders on the market and supports a variety of protocols, such as User Datagram Protocol (UDP) multicast, Real Time Streaming Protocol (RTSP), and Motion JPEG (MJPEG). ATAK can display video either full screen or half screen, the latter giving access to the 2-D map at the same time. If the video source is a UAV, and the UAV is also publishing its own position and sensor point of interest (SPI), those can be plotted on the map. Being able to see the position of the aircraft and know where on the map the camera is looking in real time, while being able to see the video on the same screen, is a huge boost to SA.

The last component that is common to any team-based activity is communications. The two most common communications methods are voice and text-based. ATAK does not have any built-in support for voice communications, mostly because the communications substrate and device (e.g., smartphone) usually has a native

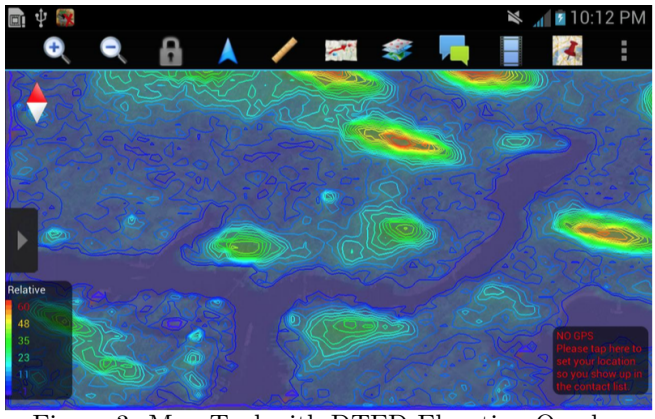


Figure 3: Map Tool with DTED Elevation Overlay

voice capability, whether it is a military radio or commercial cellular. ATAK does include a built-in chat application for text messages. Chat is a very common application, but most solutions require a server. Since ATAK was designed to be used in networks disconnected from any infrastructure, a server-less solution was needed. ATAK implemented a GeoChat (a component of the Battlefield Airmen Operations kit[3]) compatible chat client. GeoChat is a multicast-based, and requires no server. To support a wider variety of use-cases, we then extended it to be able to use point-to-point TCP in addition to the multicast-based chat.

3. ATAK FEATURES AND TOOLS

ATAK has a variety of mission planning and execution tools built in. This section discusses some of these tools and the benefits of using Android.

3.1 Android

ATAK was designed to operate on Android devices for a number of reasons. First, Android runs on a plethora of COTS, commodity mobile hardware. This variety of hardware greatly improves the likelihood of finding a platform that will meet size, weight, and power (SWaP) requirements. Second, Android is a free and open platform, making it possible to customize nearly any aspects. Third, there are a number of peripherals available including external radios, position systems, notifications, displays, etc. Finally, the software development kit (SDK) is relatively easy to use and allows for rapid development cycles.

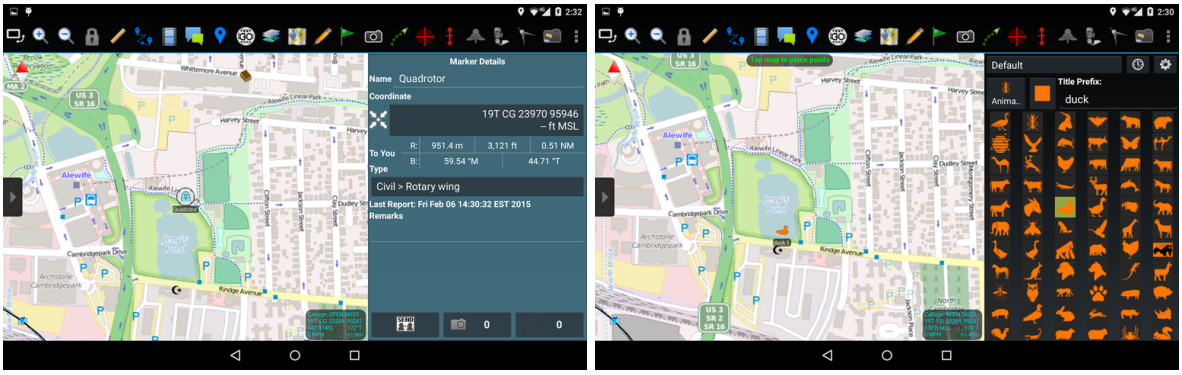
3.2 Map Tool

Of the many tools in ATAK, the most prominent is the moving map tool. It displays base georeferenced imagery and overlays other SA information (e.g., georeferenced points, annotations, etc.). The georeferenced imagery offers offline access to maps and supports many standard formats including NITF, MrSid, Precision Image (PRI), OpenStreetMaps SQLite, Web Map Tile Service (WMTS), Web Map Tile Service (WMTS), Mobile Atlas Creator (MOBAC), Gridded Reference Graphics (GRGs), and more. Additionally, the mapping engine is extensible, allowing additional map data sources to be added.

In addition to base map imagery, the map tool can display many different overlays. Some of the most used overlays include blue-force tracks, points, routes, geospatial annotation files (e.g., KML), drawing tools, and elevation (see Figure 3, e.g., DTED). Further, the map tool allows multiple map projections as well as multiple view options (e.g., north-up, track-up).

3.3 Annotation Tools

Annotation tools include those tools which allow users to contribute SA information. We distinguish these tools from planning and measurement tools in that the information that is created and visualized by annotation tools is meant to be shared with other actors, whereas information visualized by planning and measurement tools is meant to be used locally.



(a) Drop Point Tool (b) Custom Icons

Figure 4: Point/Icon Placement Tools

3.3.1 Point/Icon Placement

The point dropping tool overlays markers on the map by simply tapping on the map or entering the coordinates of a point in one of many supported formats. The icons attached to these points can include much of the MIL-STD-2525B iconography, colored spot reports, or separately-loaded custom icons (see Figure 4). Points can be named, modified, and augmented by attaching files and photos. Additionally, the updates to these points are made available via history tracking. Further, the locations of points can be entered or adjusted using precise position information. This allows users to drop points given the only their positions in various coordinate spaces (e.g., latitude and longitude, MGRS, etc.).

3.3.2 Routes and Drawing Tools

The route creation tool allows users to produce routes, essentially ordered sets of points and lets the user designate criteria about the route. For example, a user can indicate if a route was designed for walking, driving, or flying. The route tool is a special-case for a more general form of annotation, drawing tools (see Figure 5a). The drawing tools allow users to contribute georeferenced circles, rectangles, and arbitrary polylines/polygons (see Figure 5b). A survey tool helps users perform air-field, runway, and sensitive site surveys as well as other end-user-specific surveys (see Figure 5c).

Many of the annotation tools are able to be exported in geospatial annotation formats (e.g., KML), and ATAK is also capable of importing and rendering KML (including KML with network links), KMZ, CoT, and some FalconView formats.

3.4 Planning and Measurement Tools

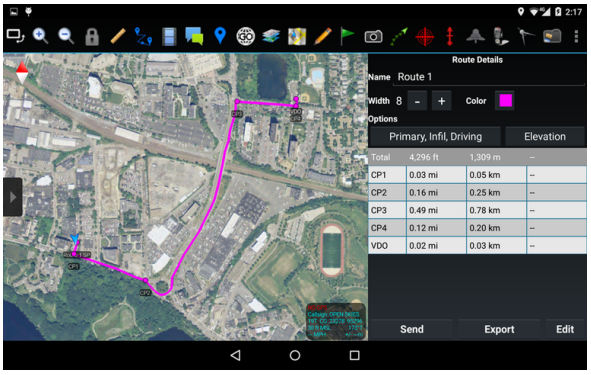
ATAK also has a variety of tools meant to help with mission planning and analysis and are designed to be used effectively before, during, and after the mission. This section discusses some of the tools with planning and measurement capabilities.

3.4.1 Range and Bearing

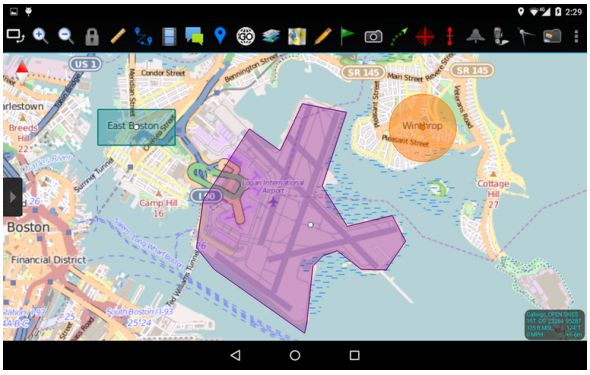
The range and bearing tool allows the user to draw persistent pairing lines between any two objects. Connecting lines and range/bearing measurements automatically update if the objects move (see Figure 6).

3.4.2 Route Planning

In addition to the standard route creation tools, there are also route planning tools that give a user more insight into certain aspects of a route, such as the elevation profile of a route (see Figures 7a). The navigation tool helps users traverse a pre-planned route (see Figure 7b). The navigation tool creates lines of bearing, centers on the user's position, and automatically selects a zoom-level, while listing your speed, distance, and bearing to the next waypoint.



(a) Route Creation Tool



(b) Drawing Tools

(c) Survey Tools

Figure 5: Annotation Tools

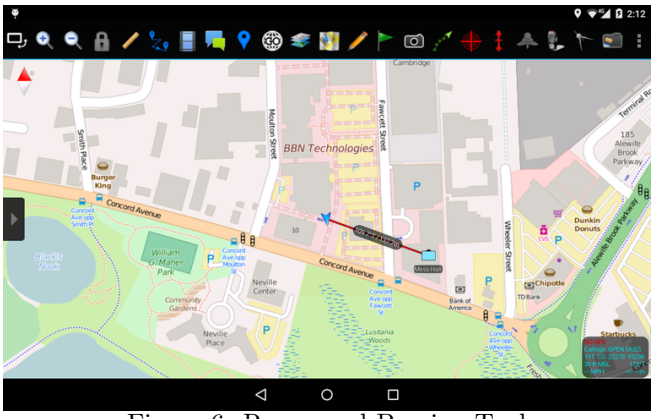


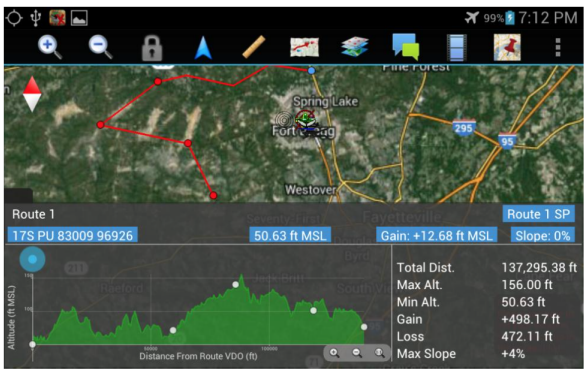
Figure 6: Range and Bearing Tool

3.4.3 JumpMaster Sky Diving

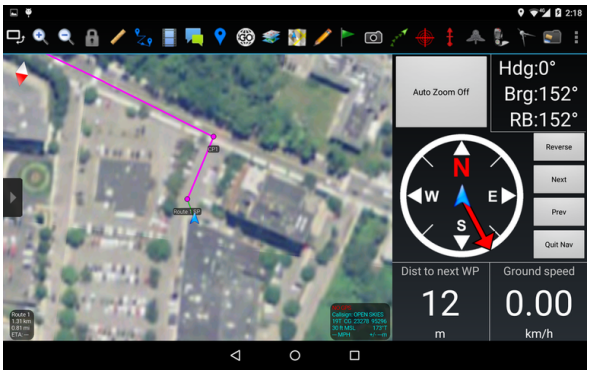
The JumpMaster tool includes a three-dimensional version of the navigation tool and helps with planning, calculating, and in-jump navigation for skydiving (see Figure 8). In addition to the features of the navigation tool, JumpMaster also includes wind calculation and auto-zoom that scales the map to the same size that is perceived by the skydiver.

3.5 Collaboration Tools

The next set of tools are designed to help users collaborate between multiple instances of ATAK, and in some cases, other mission planning, SA, or analysis tools. We examine three separate types of collaborative activities:



(a) Route Planning Tool



(b) Route Navigation Tools

Figure 7: Route Tools

Figure 8: Jump Master

(1) importing data, (2) exporting data, and (3) real-time sharing. ATAK has examples of all three of these collaborative activities. This section will discuss some of the examples.

3.5.1 External Storage Media

All of the collaborative activities require data to be moved to or from one device to another. This can be accomplished by transferring files to/from external storage media (e.g., micro SDcard), by using one of the many networked radios that exist on modern Android hardware (e.g., cellular, WiFi), or by using external networking components (e.g., tactical radios). For transferring files, micro SDcard replicators have proven very useful. By default, ATAK looks for files to import in a variety of locations so external storage media can be used.

3.5.2 Networking

For networking, ATAK works with a wide variety of commercial and tactical radios as well as WiFi and cellular which are common on COTS phones. Different networks have different restrictions, so to operate on all these types of networks, ATAK employs multiple communication paradigms. For example, cellular networks and virtual private networks (VPNs) typically prevent the sharing of multicast data. For these situations, ATAK can connect to the Marti information management system[4,5] to communicate. In situations where multicast is available, such as most tactical MANETs and many WiFi routers, ATAKs can collaborate without the Marti server.

3.5.3 Geospatial Data and File Sharing

Points, routes, drawing objects, and files can be shared through ATAK (see Figure 9). ATAK maintains a list of contacts (i.e., other ATAK instances), which is displayed when a user elects to share data. Sets of users can be aggregated in the following two ways: (1) groups and (2) teams. Groups are sets of contacts that a user

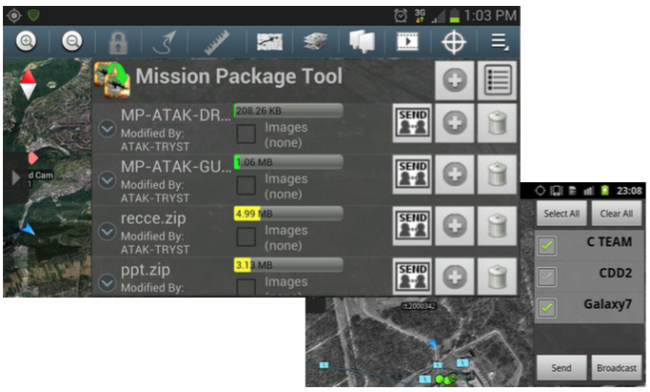
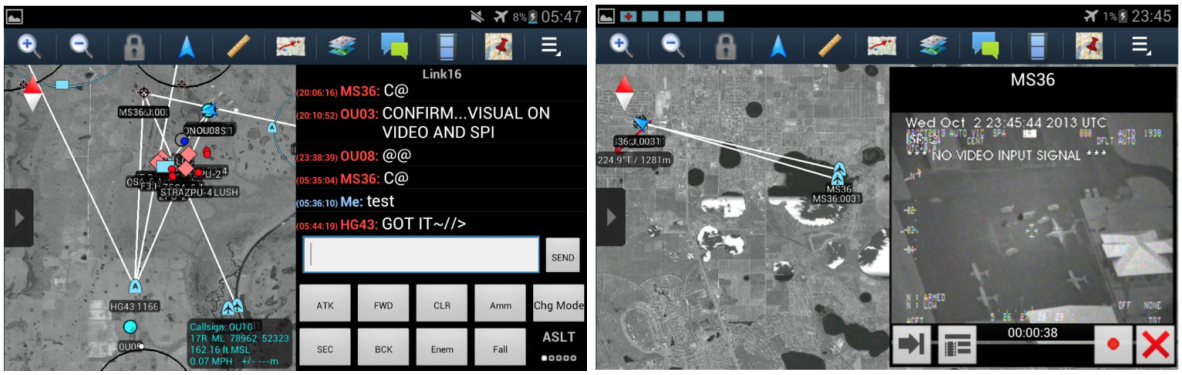


Figure 9: Point, Drawing Object, and File Sharing Tools



(a) Text Chat Tools (b) Video Tools

Figure 10: Chat and Video Collaboration Tools

designates as being related whereas teams are sets of contacts that self-identify as being related. Therefore, teams are automatically constructed from matching the team identifier (typically designated as a color) between contacts. Groups, however, by definition, must be manually created.

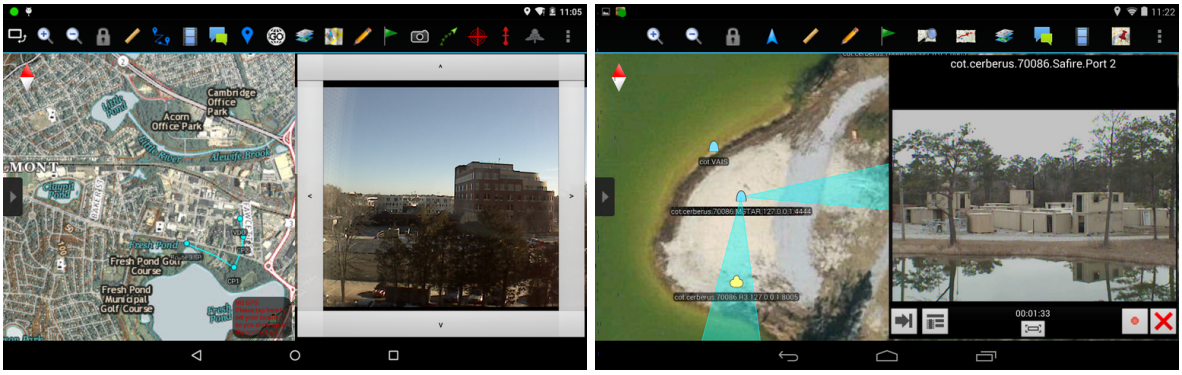
Selecting users, groups, and teams implies an information exchange with a *1:1* or *1:many* cardinality. The other option for disseminating information from ATAK is to *broadcast* it. Broadcasting is a quick and easy way to exchange information in a *1:all* cardinality, however there are some drawbacks to broadcasting. First, broadcasted information is transmitted over UDP. This means that there are fewer guarantees for the information delivery. For example, the broadcasting ATAK will not know if one ATAK user fails to receive all the broadcasted message. Also, UDP messages have an upper-limit for their size, meaning large messages (e.g., photos, videos, long routes) cannot be sent via the broadcast paradigm.

3.5.4 Text Chat

ATAK also contains text chat and presence tools (see Figure 10a). Presence is presented in ATAK's contact list. Green, yellow, and orange indicators display information about the time since ATAK last received an heartbeat message from that user. Clicking on any user, group, or team in the contact list displays a chat window that allows users to communicate via text. While most chat clients require a server, ATAK's chat feature can operate in a server-less environment. If a Marti server is available, ATAK can also utilize it for passing text chat messages.

3.5.5 Video

Another tool in ATAK is the video tool (see Figure 10b). The video tool supports a wide variety of standards-based video types/formats, including various KLV (key-length-value) metadata formats.[6] If the video contains geographic metadata, then ATAK will also plot and track the geospatial information on the moving map.



(a) Closed-Loop Video Sensor Control (b) Sensor Tasking

Figure 11: Sensor Tasking and Control

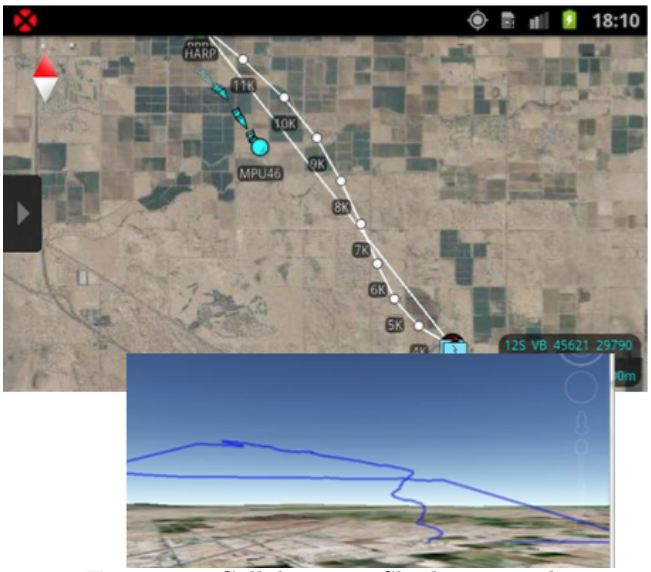


Figure 12: Collaborative Skydiving Tools

3.5.6 Sensor Tasking and Control

Related to the video tools, ATAK offers support for closed-loop video control for gimballed EO/IR sensors. That is, video sensors that can be slewed and zoomed relatively (e.g., pan 5° to the right) can be controlled directly from within ATAK (see Figure 11a). Additionally, sensors with absolute positioning APIs (e.g., slew to latitude/longitude) can also be controlled via the sensor tasking features of ATAK (see Figure 11b).

3.5.7 Peripherals

ATAK also has support for numerous peripherals, including laser range finders, notification devices (e.g., smart watches), external GPS, and more. Additional peripherals can be supported through ATAK's plugin architecture.

Additionally, many of the other tools discussed support collaborative aspects. For example, the JumpMaster tool allows for collaborative skydiving (e.g., sharing drop points and changes in trajectory). Figure 12 shows how JumpMaster can be used to help collaborate during the jump, and afterwards by exporting the jump data to KML for viewing in Google Earth.

4. EXTENSIBILITY AND CUSTOMIZABILITY THROUGH A PLUGIN ARCHITECTURE

In developing ATAK, we have kept extensibility and customizability in mind. Although ATAK has at its core a mapping application, ATAK has a growing user base, many of which need or desire features customized to their missions, operations, and specific use cases.

Rather than attempting to develop a single monolithic application that would serve all users, attempt to build different sets of customized builds that would become unwieldy to maintain, or even to attempt to have all ATAK development filter through us, we chose to produce an architecture that would enable third party developers to create customized functionality and features. As Section 3 describes, ATAK comes with a large set of useful features, many motivated by particular users and use cases. To accommodate users and developers creating additional customized features, we adopted a *plugin* architecture with an application programming interface (API) that facilitates developing plugins that can interact with the underlying core application, i.e., the mapping application, and that can interact with the ATAK and Android tools. A plugin is a software component created separately from the core ATAK application, but written to conform to the ATAK plugin API so that it can work with ATAK to add features and interoperate with the core ATAK functionality and other plugins.

This section describes the goals of (1) ATAK's plugin architecture, and (2) ATAK's individual plugins. When applicable, we explain the alternatives that were considered, the option that was selected, and the pro's and con's of our approach.

4.1 Goals of the Plugin Architecture

We have developed the extensible plugin architecture to meet the following goals:

- Facilitate Plugin Storage, Distribution, and Loading — Once a user has developed a plugin, i.e., written a software component conforming to ATAK's plugin API, there needs to be a mechanism to store the plugin, make it available to users, and load it into the application.
- Enable Dynamic Plugin Discovery and Loading — Plugins interact with the core application, and to do so, the core ATAK application must be able to discover them and dynamically load them.
- Manage Plugin Dependencies — Plugins may want or need to use features from the core application or other plugins.
- Moderate Interoperability — Plugins should be able to be used together, or be marked such that plugins providing conflicting features are not loaded at the same time.
- Manage Plugins and Configurations — There are other existing SA applications, some of which have specialized functionality different from that offered by ATAK or are missing functionality that ATAK provides. We have developed the ATAK plugin architecture to support plugins that are compatible across platforms. This includes a management scheme that selects which plugins are loaded and how they are configured in each application.
- Maintain Plugin Provenance — Plugins can be developed by third parties (i.e., not only core ATAK developers). The architecture must be able to attribute each plugin to its developers and identify trusted sources of plugins.

The following sections describe the way in which the ATAK plugin architecture meets these requirements.

4.1.1 Plugin Storage, Distribution, and Loading

This section describes the form in which plugins are stored and how they are obtained by an end user for loading.

Plugins can be installed applications or separate loadable entities (e.g., JAR, DEX, or APK files). In Android, applications that are intended to be installed are distributed as APK files.

Requiring installation of plugins can be a burden on users, especially for large numbers of plugins, since these typically have to be done one at a time on the mobile device or using the Android development tools (which are not designed for basic users) from a connected computer. For separate loadable entities, Java-based file formats are most convenient because they can be loaded at runtime using existing tools (e.g., ClassLoader, DexLoader), however, any other general-purpose or domain-specific languages can also be used, provided they can be loaded into the core application.

ATAK currently requires plugins to be installed applications, meaning that plugins are distributed as APK files. Once the plugin APK files are installed on a device along with ATAK, they will be discovered automatically by the plugin architecture and loaded.

Because plugins can be created by any third party, core ATAK developers cannot control how plugins are distributed. Plugins can be hosted by anyone or distributed by any file-sharing mechanism. Plugins that are developed by the core ATAK team are hosted on the core application distribution website.

4.1.2 Enabling Dynamic Plugin Discovery and Loading

When a plugin is needed, it can be discovered in a number of ways. The primary factor that affects the framework's discovery mechanism is the form of the plugin. By requiring that plugins be installed applications, the plugin framework has methods available to discover new plugins (e.g., asking the OS which applications handle a particular `Intent`). However, if the plugin architecture *requires* installation of a plugin, it eliminates the potential benefit of giving users the *option* to install a plugin (e.g., if it is distributed in an installable form, such as an APK).

Another benefit of requiring that plugins be installed applications is that they are capable of running services in their own process space that extend the application permissions of the core application. For example, if the core application does not require permission to modify the Android contact list, a plugin that is in the form of an installed application can declare that it requires such permission and modify the contact list from a self-spawned service.

ATAK requires that plugins be installed applications and expects there to be a particular file describing the contents of the plugin.

Plugins can either be loaded (a) into the core application's process space or (b) into a different process space. By enforcing that plugins must be loaded into a separate process space, all communication between the core application and its plugins must be through Inter-Process Communication (IPC) which can be slow and cumbersome. Loading plugins into the core application's process space allows for direct access to method calls, resources, and data at the expense of protecting the core application from the plugin's side-effects.

ATAK loads plugins into the core application space, giving plugins direct access to the application utilities made available through the API.

4.1.3 Managing Plugin Dependencies

Plugins may want to use features from (1) the core application or (2) other plugins. There are various stages at which these features can be shared or re-used (e.g., compilation or runtime). Runtime dependencies pose challenges to system maintainers such as ensuring that all dependencies are installed, all dependencies have appropriate versions, and dependency loading is sound (e.g., does not include cyclical dependencies). Compile-time dependencies pose challenges to developers, particularly in preventing conflicts between shared classes at runtime.

ATAK allows dependencies to be imposed at either runtime or compile-time. This puts the responsibility of deciding on which type of dependencies to introduce (if any) on the plugin designers.

4.1.4 Moderating Interoperability

Plugins need to share data between each other and the core application such that all features recognize data contributed from plugins. Features include displaying objects consistently, searching for items by name, location, type, meta-data.

There are likely some subset of plugins that are unable to be used together. This may be by design (i.e., they provide orthogonal features), by implementation (i.e., they require exclusive use of a shared resource), or by experience (i.e., the combination of features they provide were not logical or usable).

Currently, ATAK allows plugins to contribute geospatial annotations (e.g., icons, lines, etc.) to ATAK overlays, which makes them usable by the core application as well as other plugins. This lets ATAK tools operate on plugin-contributed items, for example, drawing range-and-bearing lines (a core ATAK tool) between icons contributed by plugins. It also allows items contributed from one plugin to be visible and accessible to other plugins.

4.1.5 Managing Plugins and Configurations

A plugin may be applicable to more than one core application. Likewise, a user may want different sets of plugins loaded at different times, i.e., a plugin configuration. Thus, the application framework must be able to manage the installation, activation, and configuration of plugins. The form of the plugin can make this particularly difficult. For instance, if the plugin architecture uses separate loadable files as plugins, then the filesystem can be used to indicate which plugins should be loaded in each configuration (e.g., put the plugin files in one directory per configuration). If, however, the plugin architecture requires that plugins be installed applications, then a separate mechanism must be used to indicate the applications or app configurations in which the plugin is intended to be loaded.

All plugins (or combinations of plugins) may not operate correctly. Thus, plugins must be able to be disabled when non-operational. The framework needs to provide a *safe mode*, i.e., a start-up configuration that does not load any plugins, for when plugins cause problems at start-up.

ATAK shares a portion of its API with another SA application, *NettWarrior*, and therefore, must manage which plugins get loaded in which application. Currently, we handle this with a configurable blacklist, i.e., a list of plugins that should not be loaded in a particular configuration. Plugins that appear on ATAK's blacklist are skipped-over during the loading process.

4.1.6 Maintaining Plugin Provenance

Plugins must be able to created and loaded by third party developers without having been vetted by the core application developers. This has implications on how plugins are distributed and loaded, but also on how their provenance is assured. The plugin framework needs a way to attribute third party plugins to a particular source. Android contains built-in utilities for signing applications such that the user can confirm that an application is provided by a known source. The Android signing tools indicate to the OS if two or more applications share the same source, and as such, should be allowed to share process-space, functionality, and data. While developed for a different purpose, the functionality is similar to what the ATAK plugin framework needs.

Since ATAK plugins may be developed by a third-party (i.e., not the core ATAK team), it is impossible to expect plugins to be signed with the core application key. If plugins were required to be signed with the core application key, then we would have to distribute the key with the plugin SDK. This would entirely defeat the purpose of using the key for provenance, as anyone could sign an "impostor" application with the core application key. Therefore, ATAK's plugin loading does not require any particular signing key to load plugins. Creating a feature to manage plugin provenance is an area of future development in the ATAK plugin framework.

4.2 Goals for Plugins

In Section 4.1, we discussed goals for ATAK's plugin architecture, i.e., the framework that allows for plugins to be safely and effectively discovered, loaded, and managed. This section, in contrast, describes the goals for individual plugins, dictated primarily by the plugin API.

ATAK's plugin API is based on the one used by the NettWarrior SA application.[7] We elected to base our plugin API on the NettWarrior API for two primary reasons. First, since NettWarrior also uses this API, plugins developed for one system will be cross-compatible with the other. Second, we were able to share some code, which helped to lessen the plugin framework development effort.

The NettWarrior plugin API is quite large, and some portions are not applicable for ATAK, so we've started by implementing a subset of the API. Additional portions of the API will be implemented in subsequent versions of ATAK, but other portions are unlikely to be implemented. Therefore, there will be some cross-compatible plugins, i.e., the exact same plugin will work in both applications.

In the first version of the API, we have several goals. Plugins should be able to accomplish a certain set of tasks, including, but not limited to, the following:

- Interacting with Map
- Interacting with Tools
- Specifying User Interface Elements
- Networking, Storage, and Sensors

4.2.1 Interacting with Map

Interacting with a moving map is of primary importance for situation awareness application plugin architectures. Interaction, in this context, has several meanings. First, plugins must be able to draw or overlay objects on the map. Second, plugins must be able to receive notifications of events on the map, such as when a user touches the map.

Drawing Techniques. There are two primary techniques for allowing plugins to draw objects on the map: overlay and contribute. The first, *overlay*, allows plugins to draw any arbitrary forms in a layer that is rendered on top of the map. This technique is also referred to as "rubber sheeting" or "bit blitting" referring to the operations in computer-aided design and low-level graphics fields respectively. The primary benefit of the overlay technique is that it is extremely generic. That is, plugins can draw any object they desire. This includes points, lines, polygons, text, icons, and many more. However, the real strength of this approach is the ability to draw semi-transparent overlays over the full screen. For example, this technique excels when drawing choropleth maps (a.k.a. heat maps) where information is encoded as a color over a geospatial area.

However, the flexibility of the overlay drawing approach comes at the cost standardization across plugins. That is, since the core application doesn't know the meanings of the different drawing objects on a plugin's layer, it cannot standardize the behaviors when users interact with them (e.g., touch handlers, scaling on zoom-in/-out, etc.). As a result, this forces plugin developers to implement more functionality themselves (e.g., collision detection, scaling behavior). Having this functionality implemented at the plugin level, rather than some higher level, will undoubtedly cause behavioral variations that will be noticed by the end-user. An option to alleviate this problem is a utility library that is distributed with the plugin development SDK, however, doing so causes the SDK to have very frequent updates — as evidenced by the core Android SDK.

Another major detriment of the overlay drawing approach is that it complicates the sharing of information between (a) the plugin and core application, and (b) the plugin and other plugins. There are several tools that require semantic information from plugins, and that semantic information is obfuscated by flattening it into a rendered drawing overlay. For example, if a plugin draws points as icons overlaid on a map, a user may expect that she can draw a range-and-bearing line to that point (as she can do with any point that is dropped within the core application). However, if the core application does not know the exact center point of that item, but only the pixels where the icon was painted, acquiring an accurate range-and-bearing to that point is impossible. While this example case is easily worked-around, it represents a class of larger and more-complex problems that arise when the semantic information around the drawing objects is lost — a property that is necessary for the benefit of drawing *generic* objects.

In addition to complicating the interactions between the plugin and core application, the drawing approach also complicates sharing information between plugins. Suppose, for instance, a new plugin is created that produces new range-and-bearing lines that display additional information. This plugin needs to be able to recognize points from other plugins, which is impossible for the same reasons that it was impossible for the core application to determine the exact point. Thus, if the overlay drawing approach is employed by the plugin architecture, the framework will also require a robust communication substrate to pass semantic information from plugins to (a) the core application and (b) other plugins.

The second plugin drawing technique, *contribute*, allows plugins to provide semantic information to the core application to be rendered as deemed appropriate by the core application. The primary difference between this approach and the overlay approach is that, in the contribute approach, the plugin yields full control of the drawing of objects to the core application. The benefits of contributing semantic information, rather than drawing primitives, to core application is that the application can provide some guarantees of behavioral consistency. For example, all icons that are drawn by plugins will be treated identically to those drawn by the core application, and other plugins also. Thus, many of the consistency problems of the overlay drawing technique are mitigated by the contribute approach.

However, the contribute drawing technique has some major drawbacks. First and most importantly, it lacks the flexibility, and therefore the extensibility, of the overlay drawing technique. That is, by definition, plugins can only contribute the types of information that are "understood" by the core application. This limits all plugins to drawing objects that are supported and exposed in the plugin API. Thus, plugin developers cannot produce objects that are unable to be drawn from compositions of existing drawing objects. As a result, it is useful when employing this technique to include an extensible object (e.g., a key-value map) in the plugin API that can give hints to the core application about how "special" items should be rendered. This allows the core application and plugins to be updated independently, but requires that the behaviors that are modified by the extensible object be well-documented.

There are also ways to combine the two approaches into *hybrid* drawing techniques. These hybrids have the capability of taking advantage of the benefits of both drawing techniques, however, they require the plugin developer to understand both approaches and when to use each. Thus, if a hybrid approach is used by the plugin architecture, it needs sufficient documentation for plugin developers to understand the implications of their plugin design decisions.

ATAK uses a hybrid technique. Currently the API supports drawing overlays using Android `Canvas` tools, or contributing items directly to ATAK's rendering engine, making them discoverable and usable by other tools.

Notification Techniques. In addition to drawing items on the map, plugins must have the ability to be notified of map-related events. Such events may include touches (i.e., the user touched the map at a certain location), scrolling, zooming, changes to the map projection (i.e., the base map layer is in a different projection than before), and more. It is best practice to make this information available via multiple paradigms if possible, so plugin developers can maintain internal consistency. That is, the plugin architecture should provide:

Paradigm	Description	Example
Accessors	methods to poll current information	`getZoomLevel()`
Callback Registration	methods to register a callback for asynchronous notifications of events	`onZoomLevelChanged(ZoomLevelChangedListener listener)`

An open question for map event notification techniques from the perspective of a plugin framework designer is the level of granularity of the events to expose in the API. That is, should plugins be passed the raw events (e.g., the unprocessed touch events on the screen in pixel positions) or the processed events that are used by the application (e.g., double-tap at a latitude and longitude). Where raw events give the plugin greater flexibility, such as the ability to monitor new event combinations (e.g., double-tap with three finger gesture), processed events allow for better consistency for the user experience. In most cases, notification techniques can make use of both raw and processed events. We have found that it is better to make the processed events easier to find

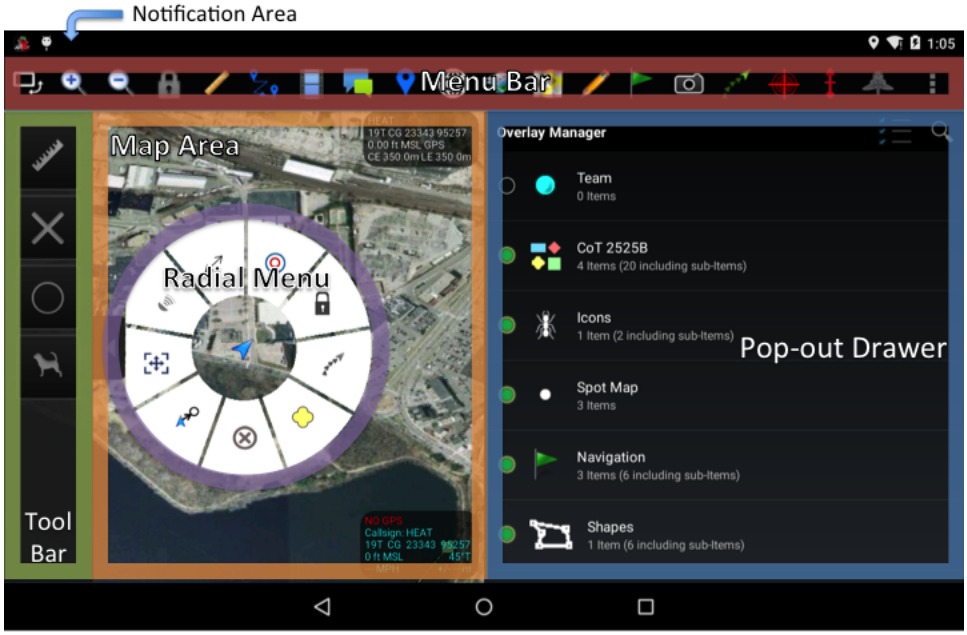


Figure 13: Components of the ATAK user interface.

and use via simple API exploration, so that plugin designers don't unnecessarily re-implement recognition for events that are already being processed.

4.2.2 Interacting with Tools

Plugins must be able to interact with the components of the core application including: pop-out drawers, toolbars, menus, settings, etc.

The core ATAK application has a number of UI components, illustrated in Figure 13. The screen area in each of these UI components will be accessible from plugins.

4.2.3 Windowing

Plugins must be able to specify GUIs that are displayed within the core application. Ideally, these GUIs components should be described in the same manner as the core application. Android has multiple methods for populating user interface components, such as inflating XML GUI descriptors and dynamically instantiating UI Views, made even more complex with the introduction of Android Fragments.

4.2.4 Networking, Storage, and Sensors

Plugins must be able to access network interfaces, persistence mechanisms, and sensor features.

Networking. There are two modalities to the plugin networking API. The first modality is *creating a new network stack*. This modality is necessary if no other plugin, nor the core application, provide the type of network stack that is required. The second modality is *piggybacking on an existing network stack*. This modality allows plugins to utilize existing messaging substrates to monitor and inject traffic that supports their data sharing objectives.

ATAK utilizes both modalities, but currently both implementations exist in a limited capacity. Plugins can utilize the first modality, maintaining their own network stack, through standard Android practices. That is,

plugins are not barred from creating their own incoming/outgoing network connections. The current limitation of this approach is that it is non-trivial to share these network connections between plugins.

CoT is ATAK's primary messaging protocol; for the second networking modality, plugins are able to piggyback on ATAK's CoT messaging infrastructure. ATAK has settings to add listeners for CoT messages on local IP/port combinations and the plugin framework allows plugins to register a listener for CoT messages that arrive on any of these network listeners. ATAK also lets both plugins and end-users configure IP/port combinations for outgoing CoT messages and the plugin framework has an API for writing to those outgoing interfaces. With the current implementation, it's difficult to control which interfaces and recipients are used when messages are written to the network, and consequently, it's impossible to receive acknowledgments that the messages are received by the intended recipient. Also, it's difficult to augment existing messages (e.g., status beacons) with information from plugins.

Plans are in-place to facilitate usage of both networking modalities. Thus, these limitations will likely be resolved in future versions of ATAK.

Storage and Retrieval. Data storage and retrieval is another common necessity for plugins. Some common use cases include, but are not limited to, the following:

- Read/write particular file types (e.g., map data, geospatial annotations, etc.)
- Persist user-selected preferences across application restarts
- CRUD spatio-temporal data for map overlay

Each of the above use-cases also requires a different method of data storage and retrieval. Reading and writing specific file types is typically accomplished via standard Java file IO and made possible by allowing read/write access to the system's SDCard through the Android permission system. Persisting preferences is accomplished using Android's `SharedPreferences`, stored in the application's data segment, and managed by the Android OS (e.g., cleared in the Android Settings). Spatio-temporal data is managed by ATAK, and stored in one of several geospatial databases. As such, geospatial data can be marked such that it re-displays automatically upon restarting ATAK. Each of these kinds of storage/retrieval can be accomplished from plugins.

One improvement that is planned for ATAK is access to all spatio-temporal data through an Android `ContentProvider`. The benefit of making ATAK's data available through the ContentProvider mechanism is that access to the data is consistent from all points, including the core application, plugins, and other applications on the same device.

Sensors. Modern Android devices have a wide variety of sensors, providing information about device position/orientation, device motion, and environmental factors.

Position and orientation information can be collected from a variety of sources. Android already offers a LocationProvider which fuses input from various sensors to provide the device's location in a variety of situations. Further, ATAK plugins are able to query ATAK's current perception of the device's location which also includes GPS from external devices or a user's dead reckoning of their own position.

Information Provided	**Example Sensor**
Absolute Location	GPS, 802.11
Relative Location	NFC
Absolute Orientation	geomagnetic field (a.k.a., digital compass)

Device motion is distinct from position and orientation in that motion sensor events are only registered when there is a change to the device's position or orientation. These kind of events are useful for monitoring device tilts, shakes, rotations, etc.

Information Provided	Example Sensor
3-axis Acceleration	accelerometer
3-axis Rotation	gyroscope
Direction-independent Motion	pedometer

Unlike the prior two categories of sensors, environmental sensors are used to monitor the environment surrounding the device rather than actions that are made to the device.

Information Provided	Example Sensor
Temperature	thermometer
Light	photoresistor
Air Pressure	barometer
Humidity	hygrometer
Sound	microphone
Optical Imagery	camera

The example sensors listed above are not meant to be a complete listing, nor is the categorization meant to be a non-overlapping taxonomy. For example, there are many of services that use optically-processed QR-codes to derive a device's location.

Additionally, there are other sensors that are not included in the Android sensor framework. For example, applications can detect events (e.g., button presses, attaching/detaching) on the device (e.g., volume button) or external to the device (e.g., on bluetooth devices, headphone jack, etc.).

The main hurdles to ensuring that all of these sensors can be accessed by plugins are (1) the API version and (2) the permissions system. Both of these are flexible in ATAK, meaning that sensors should always be accessible to ATAK plugins.

5. EXAMPLE USES

ATAK is useful in a wide variety of scenarios. The range and bearing tools can be useful even when your ATAK is completely isolated from any sort of network. For instance, when going hiking in the woods, with a couple taps on map you can drop a point on where you parked, then make a range and bearing line that is locked on your self marker and the parking spot.

However, the real value of ATAK is when you have a small group that needs to coordinate, and you have some sort of IP network. ATAK has been successfully used with a large variety of military radios, commercial cellular, and even SMS. ATAK was designed to be used in an infrastructure-less (server-less) environment while being fully functional. On the other hand, ATAK can make use of infrastructure if available. This makes it possible to have multiple network options and switch over on-demand. It also means that ATAK can operate in virtually any type of networking environment. ATAK will use multicast and direct device-to-device addressing when available. However, commercial cellular carriers often firewall off the phones (as well as not supporting multicast), so ATAK has to employ other strategies in that kind of environment.

Imagine a search and rescue scenario with a dozen or so rescuers. They could use commercial cellular 3G/4G if available, or fall back to a SATCOM or line-of-sight radio that provided an IP interface. The search team might have their devices pre-populated with a KML overlay that showed the geographic boundaries of the search areas. An operations center would watch the team members move out and conduct their search. The chat function could be used for coordination. Team members could take pictures and send them to the operations center with the geographic location and manual notes (a key note here is that the geographic location sent does not need to be the current GPS reading of the device taking the photo). If a new danger presents itself, the operations center could create a new set of overlays for the search and send them out in real-time to the team via a Mission Package. Likewise, new pictures of the victim could be pushed out to the team in real-time.

6. RELATED WORK

ATAK was conceived and continues to be developed in a climate of rapidly evolving, interrelated technologies: increasingly powerful mobile computing devices and increasingly available and rich georeferenced data. ATAK is designed to interoperate with a variety of data sources and external applications. At the same time, ATAK is only one of several SA applications being deployed by the defense, law enforcement, and intelligence communities.

6.1 Web Map Tile Service

ATAK supports downloading map data using Web Map Tile Service (WMTS). WMTS is an open standard for providing georeferenced map images from an online database.[8] It is maintained by the Open Geospatial Consortium, a worldwide organization consisting of over 500 government, commercial, and academic institutions with expertise in geographic information.

A WMTS client desiring map data submits an HTTP request specifying the region of interest and other details. The server responds with a tile (image). The tile can include overhead imagery, vector data such as elevation, political boundaries, weather, roads, or any combination of those. An important consideration with WMTS tiles is that all auxiliary data (e.g., roads) are "burned in" to the image. This means that the auxiliary data cannot be turned off (made invisible). It also means that if the map is rotated (e.g. when using the internal compass in track-up mode), the labels burned in to the tile image cannot be rotated around to always be right-side up.

ATAK supports a wide variety of WMTS servers. ATAK comes packaged with several free providers of map data, including OpenStreetMap[9], the U.S. Geological Survey (USGS),[10] and the U.S. Naval Research Laboratory (NRL).[11]

There is also a mechanism in ATAK to add your own WMTS servers. This provides immense flexibility for the cases where the open/free map sources don't have the right data or high enough resolution for your application.

6.2 Map Visualization Tools

The availability of high-quality digital maps and geospatial data drives demand for applications that can display that data for exploitation and analysis.

6.2.1 FalconView

FalconView is a GOTS mapping application designed by the Georgia Tech Research Institute that has been used operationally since 1990.[12] It is oriented toward mission planning for aviation. A version of FalconView is licensed under the open-source GNU LGPL license and can be downloaded and used free of charge.

FalconView has a prominent role in flight planning at all levels of the U.S. DoD. With respect to mobile usage, FalconView is a central component of the Battlefield Air Operations (BAO) kit. FalconView supports a wide variety of map data formats and overlay formats (such as KML, DRW, LPT), some of which were developed for FalconView and became *de facto* standards.[13] FalconView has open-source versions, extremely permissive licensing, and a Software Development Kit (SDK) for developers.

The ubiquity of FalconView for mission planning, coupled with the SDK and permissive licensing have led to a large number of third-party plugins. We hope to one day match the both the variety and sheer number of third-party plugins that FalconView enjoys.

FalconView operates only on the Windows OS. This severely limits the types of devices upon which FalconView can run. ATAK, in contrast, runs on Android devices, which have a variety of form factors, many of which are lighter, smaller, and have superior battery life compared with even the smallest Windows laptop. The form factor of devices capable of running ATAK allows for many unique features, e.g., JumpMaster which is designed to run chest-mounted during skydiving.

Furthermore, FalconView's main component is the map engine, and its original use case was mission planning. It lacks integrated communication features (e.g., chat, video, etc.), requiring the user to employ additional software to accomplish those tasks. ATAK integrated tactical communications features into the SA app, providing a more fluid user experience and enhanced data sharing.

6.2.2 Esri

Esri is a US-based company that sells geospatial information systems and data services. ArcGIS is their proprietary software platform for viewing, analyzing, and editing geospatial data. ArcGIS is desktop and server software, and many of the products ArcGIS outputs (including SHP files) can be used by ATAK.

Esri also has an Android toolkit. The AGC trade study mentioned earlier [2] compared ATAK to the Esri Android toolkit. They found that the support for native map product types was effectively equivalent.

6.2.3 Google Earth

Google Earth is a powerful geospatial visualization. The Google Earth application is accompanied by a data service that provides low-resolution map data free of charge.

Google Earth is one alternative for mobile situation awareness. It has versions that are designed for use on desktops/laptops, in a browser, and on mobile phones. For all of these versions, Google Earth has very robust and fast 3D rendering of map, satellite, or hybrid map data. However, only a small amount of map data can be cached to be used offline, and the user has very limited ability to control what map data is cached.

Further, Google Earth's primary interface for visualizing data is KML. It has extremely rich KML visualization, however, KML as a protocol lacks the expressiveness to perform several things that are necessary during operation execution. Of them, asynchronous updates is probably the most important. KML can only be polled from a server, meaning that data cannot be refreshed more often than the polling interval: out-of-date information on an SA display can be dangerously misleading. Also, this polling paradigm does not scale well since every client must make individual, nearly identical requests.

ATAK also renders KML and often offers users the option to import/export data to/from KML. However, to supplement KML's deficiencies, ATAK also offers a number of methods for users to contribute messages/notifications asynchronously. As such, Google Earth functions well as a pre-planning or post-mission situation awareness tool, but lacks the features necessary to serve as an execution tool.

Google offers a number of other products that, at first glance, appear to be similar to ATAK's features. For example, Google Hangouts allows for chat, Google Maps offers navigation, and YouTube has video streaming. However, all these services lack the ability to operate effectively when away from Internet connectivity. We find that operation in austere environments is very often a requirement for situation awareness applications, and therefore, ATAK is designed to operate without reachback of any kind (i.e., it is server-less).

6.3 Mobile SA Applications

ATAK is only one of several handheld SA applications being deployed by the US Government. Other major SA applications include Trans Apps, Nett Warrior, and FalconView.

6.3.1 Trans Apps

Trans Apps (short for "transformative applications") is a US Defense Advanced Research Projects Agency (DARPA) program to develop mobile applications for warfighters. Like ATAK, the Trans Apps program adopts a rapid acquisition and deployment model and utilizes commercial off-the-shelf Android hardware.

Trans Apps differs from ATAK in that its aim is to create a platform and marketplace for a diversity of independent applications. ATAK, in contrast, is a single, integrated application. There are over 50 different apps in the Trans Apps marketplace.

Trans Apps uses a modified version of the Android operating system with enhanced security features. ATAK does not require any modification to the Android OS, so it can run either on the commercial-off-the-shelf OS or on the secure Trans Apps variant.

The Trans Apps platform relies on military radios. ATAK can use IP radio networks and can also operate over commercial LTE networks.

6.3.2 Nett Warrior

Nett Warrior is currently the US Army's program of record for mobile situation awareness applications. It is designed for use by tactical leaders and is currently fielded by Army Rangers and the 10th Mountain Division.[14]

The Nett Warrior end-user device shares many of the same features as ATAK. It is based on a real-time map display and runs on off-the-shelf Android devices. However, the scope of NW is much larger than ATAK. NW provides not only a mapping application, but a base Android OS, hardware support, and a variety of auxiliary applications. While this makes ATAK lower overhead to adopt, since you can run it on most stock Android devices, NW can meet more stringent security requirements. For instance, ATAK cannot enforce lock-screen policies (screen timeouts, passwords, etc). NW on the other hand can reach into those "OS"-level functions in part because they provide their own OS.

In October 2014, the Nett Warrior program completed a trade study to select a map engine for the Nett Warrior End User Device. The study recommended ATAK's map engine be integrated into Nett Warrior.[2] Nett Warrior has already adopted ATAK's map rendering engine

Nett Warrior is designed to operate over military radio networks instead of commercial LTE networks. ATAK supports both military and commercial networks.

7. CONCLUSIONS

ATAK provides a wide variety of geo-spatial and communication tools that have proven themselves in both military and civilian contexts. ATAK has a large developer community that supports the various tools and users. Situational awareness is a common problem; many of the tools in ATAK are equally applicable to groups of hikers as they are to squads of soldiers. Moreover, ATAK has been used with communication substrates that are common in both the military and commercial sectors. Any radio that provides an IP layer, from a military tactical mesh network to commercial 3G/4G cellular, can and has been used by ATAK to coordinate both with other ATAK devices and with other tools on the network.

ATAK's support for various geo-referenced map formats is a critical issue, because we have learned that, in practice, users will get map data from any possible sources, and the tools for converting map data between formats are error-prone and not widespread. ATAK has many tools built-in, but there will always be capabilities that are particular to a specific jobs or workflows. The plugin capability addresses this issue, and allows ATAK to be the basis of an even greater array of geo-spatial SA tools.

We anticipate wide adoption of ATAK in the military, law enforcement, and first-responder communities. A civilian-oriented version of ATAK is currently being produced. Our hope is that the civilian version will encourage additional plugin development, producing useful tools that we have yet to envision and a greater suite of capabilities for users.

ACKNOWLEDGMENTS

The authors would like to thank both the ATAK developer community and the funding for development and transition of ATAK provided by the U.S. Department of Defense. We would also like to thank the endless stream of users who have provided feedback and direction. Their patience in testing and their skill in pushing ATAK to its limits have been invaluable.

REFERENCES

[1] Dominguez, C., "Can SA be defined?," in [*Situation Awareness: Papers and Annotated Bibliography*], 5–15, Air Force Materiel Command, Wright-Patterson Air Force Base, Ohio (1994). `http://www.dtic.mil/get-tr-doc/pdf?AD=ADA284752`.

[2] Leonard, K. and Cray, D., [*Nett Warrior Map Engine Trade Study*], Army Geospatial Center Systems Acquisition Support Directorate, Alexandria, VA (October 2014). `https://upload.wikimedia.org/wikipedia/commons/d/dc/US_Army%27s_Nett_Warrior_Map_Engine_Trade_Study_Report.pdf`.

[3] Eshel, T., "AFSOC to equip dismounted teams with an advanced wearable c4," *Defense Update* . http://defense-update.com/20120626_afsoc-to-equip-dismounted-teams-with-an-advanced-wearable-c4.html.

[4] Gillen, M., Loyall, J. P., and Sterling, J., "Dynamic quality of service management for multicast tactical communications," in [*Proc. of the 14th IEEE Computer Society Symposium on Object/Component/Service-oriented Real-time Distributed Computing (ISORC)*], (March 2011).

[5] Gillen, M., Loyall, J., Usbeck, K., Hanlon, K., Scally, A., Sterling, J., Newkirk, R., and Kohler, R., "Beyond line-of-sight information dissemination for force protection," in [*Proc. of the Military Communications Conference (MILCOM)*], (October 29-November 1 2012).

[6] "Motion imagery standards board: Klv metadata dictionary." http://www.gwg.nga.mil/misb/stdpubs.html (accessed March 2015).

[7] Rosen, J. L. and Walsh, J. W., "The nett warrior system: a case study for the acquisition of soldier systems," tech. rep., DTIC Document (2011).

[8] "Web map tile service." http://www.opengeospatial.org/standards/wmts (accessed January 2015).

[9] "Open street map." http://www.openstreetmap.org/ (accessed January 2015).

[10] "U.S. geological survey: Maps, imagery, and publications." http://www.usgs.gov/pubprod (accessed March 2015).

[11] "U.S. naval research laboratory geospatial computing tile server." http://geoint.nrlssc.navy.mil/ (accessed March 2015).

[12] Bailey, C., "Department of defense use of falconview." http://www.blm.gov/style/medialib/blm/nifc/aviation/airspace.Par.77886.File.dat/FalconView.pdf (accessed January 2015).

[13] "Falconview feature matrix." https://www.falconview.org/trac/FalconView/wiki/FeatureMatrix (accessed January 2015).

[14] Dixon, A. and Henning, J., "Nett warrior gets new end-user device," (June 2013).