Tango Sim Specification (draft)

1.0 Introduction

This document outlines an minimum framework for building a CIV C2 demo using Tango as communication backbone and an simulation engine as a event generator. It is intended to be used within the Tango/CIV group as a guideline to rapidly implement a C2 demo prototype that can be further enhanced and extended, based on our gradual understanding and continuing clarification of the C2 scenario. Key approaches and components are described, including:

1) Simulation and C2 Demo Architetcure

2) Minimum C2 participates and their GUIs

3) A simulation engine and a simulation scripting language

4) Tango implementation

2.0 Goals

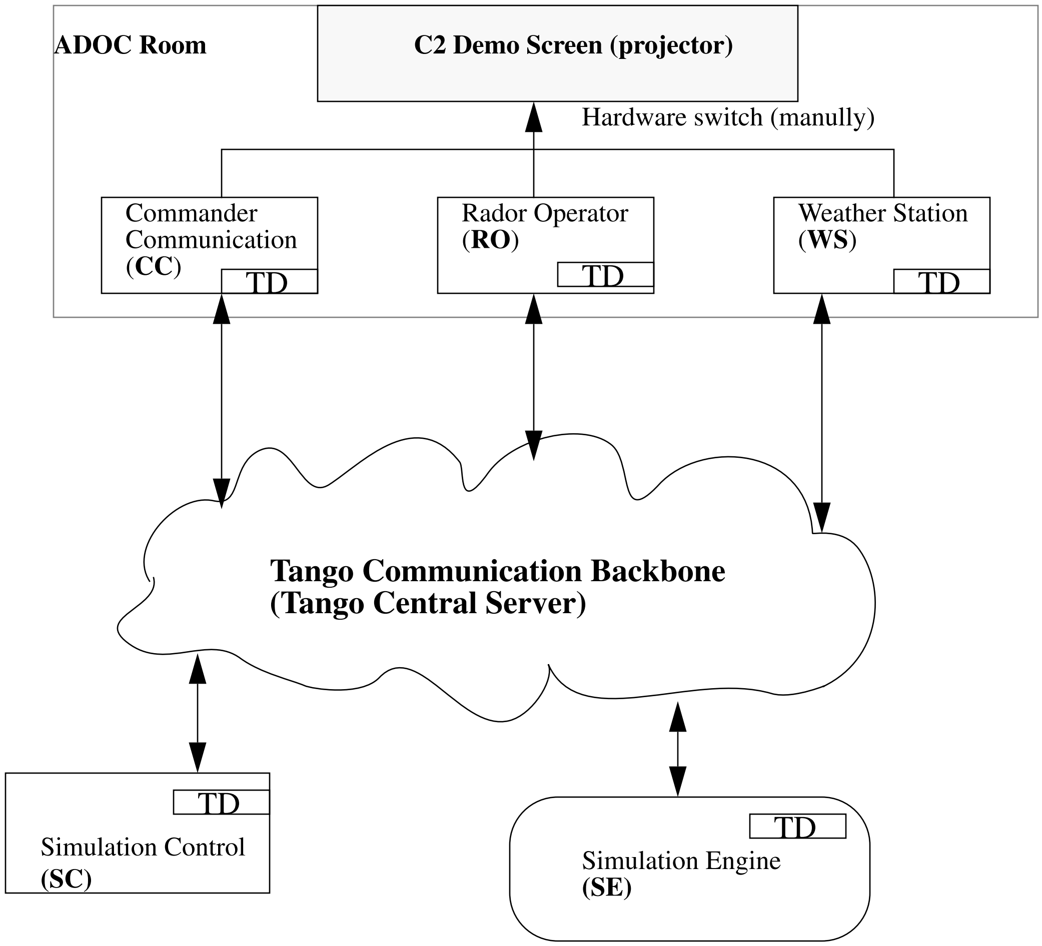
This framework will achieve the following major goals:

- Serve as a starting point to build sophisticated C2 demos that ultilize the Tango communication system
- Define a simulation infrastructure that is general enough to accomadate C2 scenario to be further detailed and scripted. It is also intended to be a test-bed for protoyping C2 demos with different scenario descriptions
- Propose a implementation guideline that minimizes the modification of current Tango system, in particular, server management, communication protocol, user management and session control in Tango

3.0 Tango Sim Architecture

We propose an architetcure that has an minimal number of C2 participates and uses Tango as communication subsystem for message passing and event routing among participates. This architecture should be easily extended to include additional C2 participates required in the C2 demo scenario, without any change of the architetcure and its exisiting components and protocols. For the time-being, we identify one key C2 demo user, ADOC consisting of three horozental ADOC participates, all other C2 demo participates are represented in the scripting text file, simulatiing their presence and response to events.

FIGURE 1. Tango Sim Architecture



3.1 ADOC Room

ADOC consists of three individual workstations, each of which is connected to a big projector screen. ADOC is logically a single unit in the C2 demo, to be used by ADOC commander. The three workstations are supposed to be placed in the same room and the swiching among them is done via manually switching the connection between the projector and one of the workstaion screen. Each workstation may run a different GUI, based on the role and function it plays in ADOC. There will be no networking communication among the three workstation, although logically they reprsent different aspects of the ADOC.

Based on the current demo Vanguard developed, ADOC as a whole has three basic logical function parts:

1. GIS

Foreground: Showing radar, aircrafts and their tracking

Background: Showing 2D geographic map for the concerned regions

2. Tables

A place to list in tabular format data of concerns to well informed the changing situation of war game, to gather whatever textual information from other sources etc.

3. Directives

Widgets to issue/direct commands.

Each of the three workstations focuses/details on one or more of the basic functions mentioned above, using the limited scenario we have, ie., radar/aircrift flying through a region.

1) Commander Communication (CC)

Key functions:

o facility/GUI to establish comunication channels with other C2 participates such as NORA, FAA,etc, and outside ADOC

o Tool sets of all collaboration programs, including chat, audio tool, video,email

o focus on directives and tables to issue/send orders, and interact with other C2 participates to gather data for making decisions or simply follow orders from higher authority

o no GIS screen

2) Rador Operator (RO)

o focus on GIS display with graphical representation of radars, aircrifts and their dynamica trackings

o allow zoomings of background GIS map, while keeping the coordincated/adjusted foreground targets

o extensible to allow overlap of multiple GIS map layers

o table data about radars, aircrifts and their tracking positions

o no directives

3) Weather Station (WS)

o focus on GIS/Weather layers

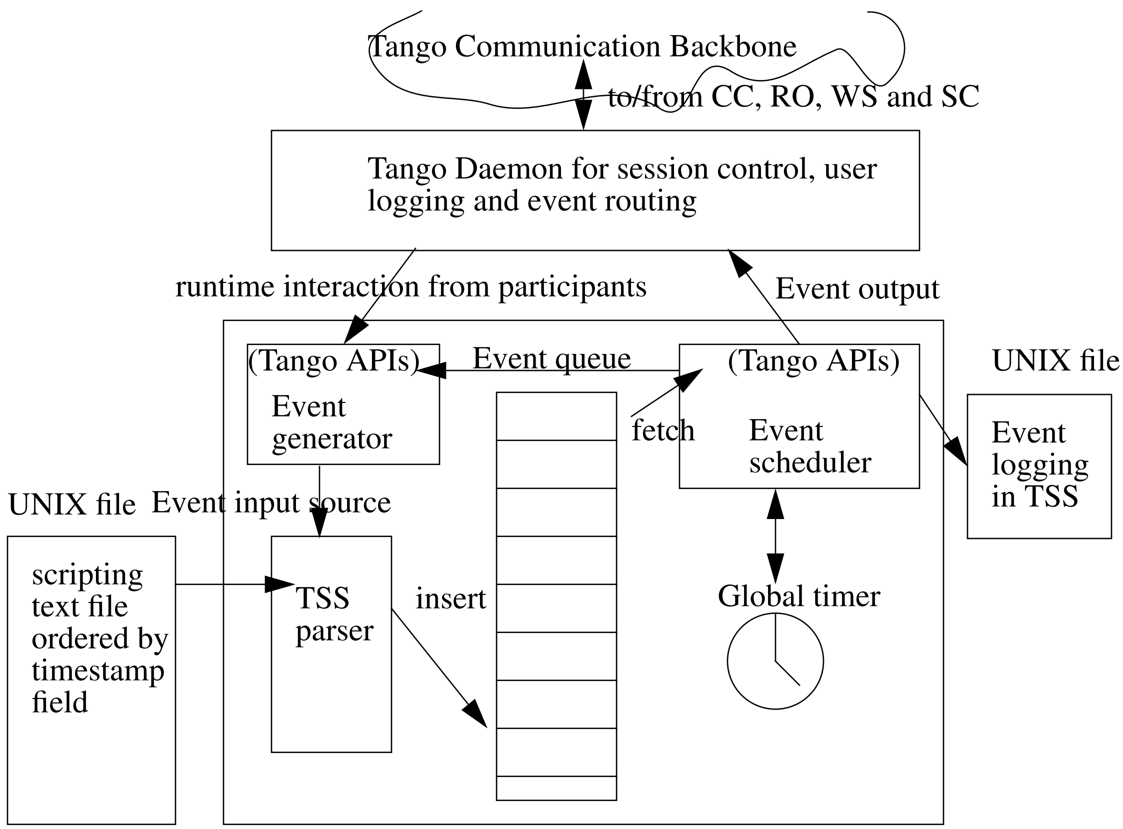
o table data about weather conditions

o no directives

3.2 Simulation Engine (SE)

SE is the center picece of Tango Sim. It is the source of all events generated, either from a predefine scripting text file, or at runtime from the C2 participate interactions.

FIGURE 2. The SE Structure and Components



Key components in SE include:

1. Tango Simulation Script Language (TSS)

A simple text language to be used to define events, messages,and protocols to simulate the C2 demo scenario that will drive the GUI behaviour on all the C2 participates. It is also used to simulate any non-existing C2 participates and their responses to events.

The baisc unit of a TSS statement is a line, ended with a newline (‘\n’). All TSS lines have the same format:

timestamp,application name[&application name],simulation id,message body of variable length

where

timestamp: represents the relative time in second to the time when the overall simulation is started. Ie, 0 is the starting time. For example, 15 means 15 seconds since the simulation started. In implementation, one integer is used to hold this field.

application name: the name of an predefined Tango session of an application. One or more application names can be supplied, which means this event will be sent to all the application sessions specified, all at once, although in reality they are sent in the order specified in this field. Note '&' is used as delimiter to indicate multiple recipents. If an message is sent to all the recipents known to the SE, a wildcard '*' can be used to subsitiute the application field.

simulation id(SID): SID is used to uniquely identify a message to be interpreted by its recipant(s). Because it is possible that the same message may have multiple destinations, 1-1000 is reserved for multiple sessions to share the same id. 1001 - is for a single application. Note the division of the SID range is for convinience only. Applications can still override this convention by mixing two together. For example:

15,SE-RO&SE-WS,20,<message1>

can be written in two lines with different SIDs:

15,SE_RO,20,<message1>

15,SE_WS,1020,<message1>

Because it is up to the recipents to decide how to interpret the SID, as long as SIDs are unique to a recipent, there will be no conflict. For description concrete purpose, the first convention is recommanded.

message body: message specific to the application(s)

Note ',','&','*' are reserved keywords (delimiliters) by TSS. They should not occur in the application name or message body. But they can be allowed in the message body.

Here are some examples of the TSS scripts:

1,SE-CC,1000,<msg body1>

2,SE-RO,1000,<msg body2>

10,SE-CC&SE_RO,1,<msg body3>

15,SE-CC,2,<msg body4>

15,SE-WS,2,<msg body4>

20,SE_RO,1001,<msg body5>

20,SE_WS,500,<msg body5>

30,*,5,<msg body6>

The syntax of TSS should be very simple to allow efficient parsing, filtering and to be converted into internal representation. Actually, it is desirable that the implemntation of SE keeps the exact same format when an event is stored on the event queue, as well as when the actuall message to be sent through Tango.

2. Event queue

Event queue is the data structure to store discrete events to be scheduled to drive the simulation. It is a dynamic prority queue sorted by the timestamp key of an event. There are two input sources: initially a static queue is established by reading from a TSS text file; during runtime, this queue may be re-sorted/organized when new events generated from an event generator are inserted.

3. TSS parser

It is used to error-check TSS source and if necessary convert into compact representation. Because the syntax of TSS is simple, the parsing should be straight-forward and efficient. Because some flexibility is allowed in using TSS to specify events, some conversion/filtering may be required. For example:

t1,a1,sid1,msg1

t1,a2,sid1,msg1

should be converted into the more efficient yet equivlent representation:

t1,a1&a2,sid1,msg1

or vise versa

Note the parser takes input from both a text file and the runtime event generator, assuming that the generator will not do type-checking/parsing, although it is expected that the events generated by the generator are always correct.

4. Event generator

It acts in two fundemental roles: for existing participates (such as CC,SC), it recieves their events over Tango that represent the runtime response/interaction so that the simulation has some dynamic alternatives to be interesting; for simulated participates such as (FAA,NCA), it may generate predefined events. For simplicity, the initial implementation will not consider generating predefined events for simulation participants. Instead, we will use scripting text file to describe any predefined/fixed events from non-existing participants.

5. A global simulation clock

It is used to schedule event output that will drive the overall simualtion. Among all the participants, only one single global clock is kept in SE. There is no local clock on participant machine. This clock can be implemented as a counter in second, based on the timestamp used in the TSS event format. For example, to schedule a new event to be sent, first the timestamp of the front event in the queue is fecthed into the clock (clock reset), then the clock starts reverse-counting (--) in every second, when a zero is reached, it activates the event scheduler (thread) to send the event.

6. Event scheduler

By looking into the application name(s) of an event, it finds out the actual Tango session as recipent. Tango session<->Application name is kept in the Tango daemon. It then uses Tango API to send an event over the network. It is possible there can be more than one events with the same timestamp but different applicaiton names or messages. So the event scheduler may need to scan the queue to check if more than one event must be scheduled at the same time. Event scheduler may also log all the events eventually sent into a UNIX log file in TSS fomat to record the overall simulation process. It is also possible that this log file be reused later to reply the simulation.

7. Tango Daemon

This is a customized Tango daemon in Java whole role is to establish predefined/runtime sessions for SE and to carry the actual message passing between SE and other C2 participants. All the Tango API and routing protocol (between TD and CS) should keep unchanged. Some changes must be made to establish the communication channels. For example, starting a new session of an application may be different for SE as it is not necessary to invoke an application applet/program, while the ‘virtual’ session must be established in which SE is always the ‘Master’. Details of using current Tango session mechanisms to form communication channels will be described later.

3.3 Simulation Control (SC)

SC has three key functions:

1. For minimal Tango Sim administration over the network, it monitors C2 participants, sessions, etc in usual Tango CA interface. Whenever a new Tango session is established, SC should be updated.

2. display simulaton progress in textual/tabular formats, show events for non-exisiting (simulated or not-connected) participants and for exisitng connected participants.

3. Control the starting, stop, pause of the simulation engine. For simplicity, initial demo only implements ‘start’ control. When the SE stops, it will be notified.

It is assumed that a ‘root’ user will use SC to control and display SE progress. One purpose is to usee SC to describe/show the demo events which have no GUI effects or are difficult to demonstrate via a computer screen. It should also leave the room for future enhancement to include more meaning display screen of the overall simulation progress and history.

This is a behind-the-scene screen that is similar to the event logging in SE but with much more readable/meaningful textual description of each event in the simulation.

4.0 Implementation Outline Over Tango

4.1 Tango Sessions

To keep the exisiting Tango framework least modified, each C2 participant (CC,WS,RO) has a session established in which there will be two Tango participants: the C2 participant, SE, in which SE is the default Tango master. It is assumed that only one Tango application (applet) runs on each C2 participant workstation that will interact with the SE during the course of simulation. Other applets are local to the C2 workstation. Actually, the Tango applet may be a ‘faceless’ Tango CA that is customized for each C2 workstation.

There is a special session with only SC and SE as participants. It runs a master-master mode so that SC will start the SE and event logging will be directed from SE to SC for meaningful interpretation.

For each event sent to a C2 participant, SE must send it to SC via the SE-SC session at the same time. Also for each runtime event received by SE from any C2 participant, it will also be sent to SC by SE.

Suppose we use SE-”C2 participant name” to represnt a session, Figure 3 shows the minimal sessions to be established for event communication.

4.2 Procedures to establish Tango Sessions

FIGURE 3. Tango Sessions