1. System Description

This assignment will model an Assault Rifle, similar to those which are in service with many military forces worldwide. This specific model is designed after a gas-operated, air-cooled, magazine-fed weapon, which would fire a variety of NATO Standard 5.56mm bullets. More details can be found in the References Section.

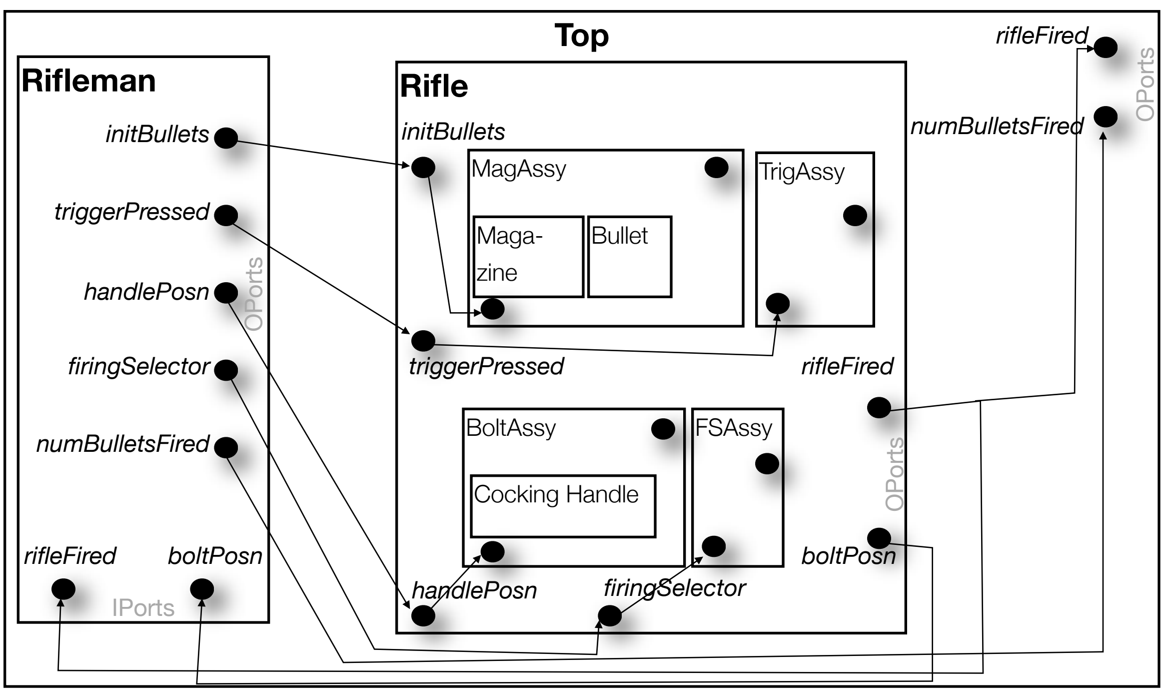
2. Assumptions and Limitations

It is assumed that a basic model of the Assault Rifle will suffice for the analysis of this problem. Details of atmospheric effects, long term barrel wear, sub-sub-component life-cycle, human response to various stimuli (fatigue, weather), will not be modelled explicitly. Furthermore, the focus of this model is the sequencing of the mechanical logic of the system, and not the exact timings themselves (ex. how long it takes a bullet to be chambered). As such, the timings for each sub-assembly are approximated, and can be readily changed to represent an actual system, should further study warrant that level of detail. This model conforms to the basic sequence of mechanical events, within the interior of an assault rifle.

3. System Model Sketches

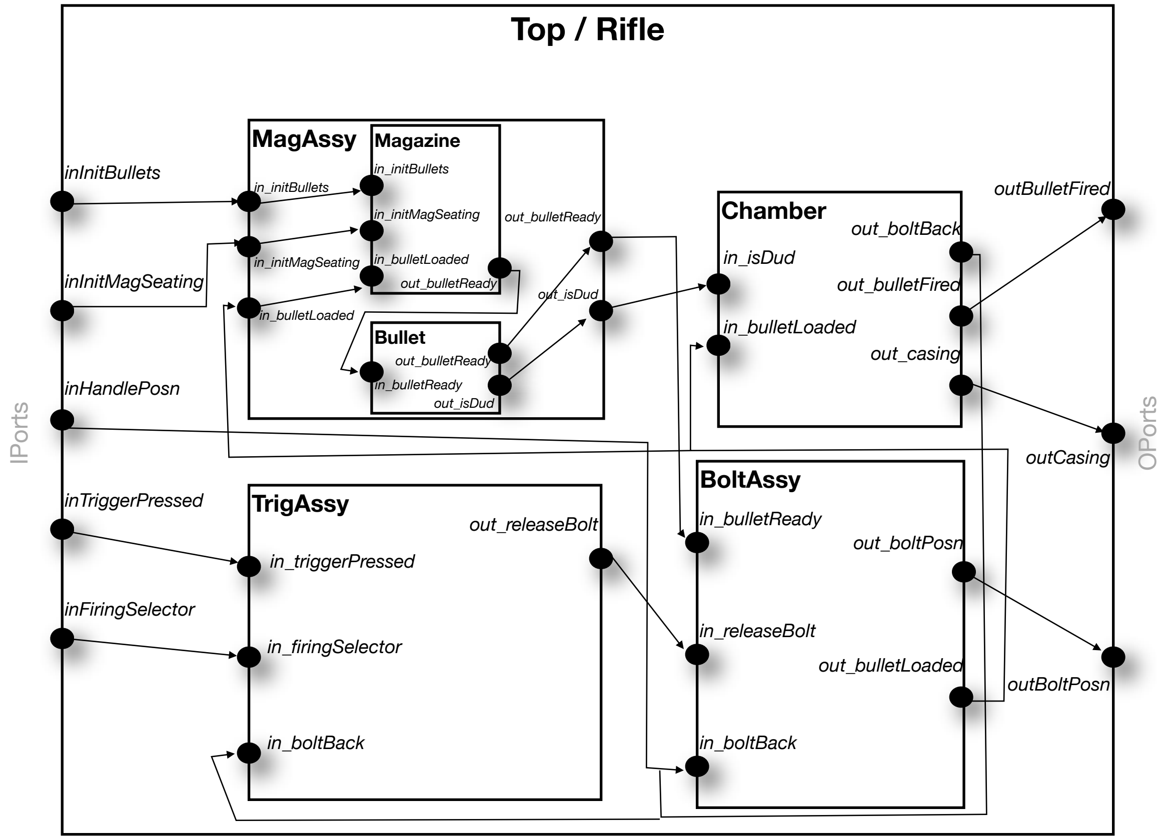
3.1. Original Model Sketch

The Rifle system was originally modelled to interact with a Rifleman model, which would be able to respond appropriately to outputs for the Rifle (blockages, poor Magazine Seating), and similarly give appropriate inputs to the Rifle. For time and simplicity, this assignment focuses on the Rifle itself.



3.2. Revised / Detail Model Sketch

A detailed Rifleman model was not implemented for this assignment. The following is a revised model sketch, including internal Atomic model linkages:



4. Description and Formal Specification

The models for this Rifle are described in the hierarchical order, by which DEVS is structured - starting with the top / root Coupled Model of the Rifle, and describing each sub-component in turn, down to each Atomic Model specification. Each component/sub-component will be described, after which it will be specified.

Rifle Description. This Assault Rifle is composed of a Magazine Assembly (MagAssy), a Trigger Assembly (TrigAssy), a Bolt Assembly (BoltAssy), and the Chamber. The Rifle is initially loaded with a magazine, with up to 30 bullets (inInitBullets). The magazine can be seated properly, or not (inInitMagSeating), which will affect the operation of the system. To begin operation of the weapon, a cocking handle must be initially pulled back (inHandlePosn), to bring the bolt back. Lastly, the firing mode must be selected, which will either be in Safe mode, Single Shot mode, or Automatic mode (inFiringSelector). When the weapon is successfully operated, a bullet is fired in each cycle (outBulletFired), a brass shell casing is ejected (outCasing), and the bolt's position must be noted (outBoltPosn) - this is of concern for the Rifleman who would be operating the weapon, to rectify any faults in its operation.

Rifle Specification. The following is a formal specification of the Rifle, using the DEVS Formalism:

M_{Rifle} = <X, Y, D, {M_d | $d \in D$}, EIC, EOC, IC, select>, where

X = {(inInitBullets, {$\boldsymbol{N}_{0..30}$}); (inInitMagSeating, {0,1}); (inHandlePosn, {0,1}); (inTriggerPressed, {0,1}); (inFiringSelector, {0,1,2})};

Y = {(outBulletFired, {1}); (outCasing, {1}); (outBoltPosn, {0,1,2})};

D = {MagAssy, TrigAssy, BoltAssy, Chamber};

M_d = {$M_{MagAssy}$, $M_{TrigAssy}$, $M_{BoltAssy}$, $M_{Chamber}$};

EIC = {((Rifle, inInitBullets), (MagAssy, in_initBullets)); ((Rifle, inInitMagSeating), (MagAssy, in_initMagSeating)); ((Rifle, inHandlePosn), (BoltAssy, in_boltBack)); ((Rifle, inTriggerPressed), (TrigAssy, in_triggerPressed)); ((Rifle, inFiringSelector), (TrigAssy, in_firingSelector))};

EOC ⊆ {((Chamber, out_bulletFired), (Rifle, outBulletFired)); ((Chamber, out_casing), (Rifle, outCasing)); (BoltAssy, out_boltPosn), (Rifle, outBoltPosn))};

IC ⊆ {((MagAssy, out_bulletReady), (BoltAssy, in_bulletReady)); ((MagAssy, out_isDud), (Chamber, in_isDud)); (TrigAssy, out_releaseBolt), (BoltAssy, in_releaseBolt)); ((BoltAssy, out_bulletLoaded), (Chamber, in_bulletLoaded)); ((BoltAssy, out_bulletLoaded), (MagAssy, in_bulletLoaded)); ((Chamber, out_boltBack), (BoltAssy, in_boltBack)); (Chamber, out_boltBack), (TrigAssy, in_boltBack))}; and

select = {Chamber, BoltAssy, TrigAssy, MagAssy}.

4.1. Magazine Assembly Coupled Model

Magazine Assembly Description. The Magazine Assembly (MagAssy) encompasses the Magazine (Magazine), and the bullets therein (Bullet). The system's model was designed this way, to allow for possible future expansion of the bullet's detailed dud rate. The Magazine may hold up to 30 bullets. When a bullet is loaded by the bolt, there is a feedback loop to the MagAssy to decrease the number of bullets. A fictitious port to communicate dud rate was designed in, to allow for the notion of a bullet having a dud rate, which is decided by a bullet atomic model, but is actioned upon in the Chamber.

Magazine Assembly Specification. The following is a formal specification of the Magazine Assembly, using the DEVS Formalism:

$M_{MagAssy}$ = <X, Y, D, {M_d | d ∈ D}, EIC, EOC, IC, select>, where

X = {(in_initBullets, {$\boldsymbol{N}_{0..30}$}); (in_initMagSeating, {0,1}); (in_bulletLoaded, {0,1})};

Y = {(out_bulletReady, {0,1}); (out_isDud, {0,1})};

D = {Magazine, Bullet};

M_d = {$M_{Magazine}$, M_{Bullet}};

EIC = {((MagAssy, in_initBullets), (Magazine, in_initBullets)); ((MagAssy, in_initMagSeating), (Magazine, in_initMagSeating)); ((MagAssy, in_bulletLoaded), (Magazine, in_bulletLoaded))};

EOC ⊆ {((Bullet, out_bulletReady), (MagAssy, out_bulletReady)); ((Bullet, out_isDud), (MagAssy, out_isDud))};

IC ⊆ {((Magazine, out_bulletReady), (Bullet, in_bulletReady))}; and

select = {Magazine, Bullet}.

4.1.1. Magazine Atomic Model.

Magazine Description. This Magazine takes inputs on the initial number of bullets, the nature of the magazine seating, and if a bullet has been recently loaded. From this information, it determines if there is a bullet at the top of the magazine which is ready to be loaded by the bolt (bulletReady). As a construct of model operability, it outputs a bullet being ready to the bullet model, which will in turn process this information (see below). Of note, since the Magazine is a near-static entity, time with which it takes to process information was set to 0 (preparationTime = 0). However, the model has the flexibility to incorporate a more precise estimation of Magazine mechanics, in a future development of this model.

Magazine Specification. The following is a formal specification of the Magazine, using the DEVS Formalism:

$M_{Magazine} = <X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta>$, where

X = {(in_initBullets, {$\boldsymbol{N}_{0..30}$}); (in_initMagSeating, {0,1}); (in_bulletLoaded, {0,1})};

S = {state ∈ {passive, active}, preparationTime, bulletsLeft ∈ {$\boldsymbol{N}_{0..30}$}, magSeating ∈ {0,1}, bulletReady ∈ {0,1}}

Y = {(out_bulletReady, {0,1})};

```
δint (s) {
        passivate();
}

δext (s,e,x) {
        if (port == in_initBullets)
            check if it is a valid amount, between 0 and 30 and if so,
            assign bulletsLeft to that value
        else if (port == in_initMagSeating)
            assign magSeating to this value
        else if (port == in_bulletLoaded)
            if bullet is actually loaded (=1) decrease bulletsLeft by one
        state = active;
        ta (state) = preparationTime;
}

λ (s) {
        if bulletsLeft >= 0, and the magazine is seated, then a bullet is ready
        else bullet is not ready
        sendOutput(time, out_bulletReady, bulletReady)
}

ta (state) = preparationTime;
```

4.1.2. Bullet Atomic Model

Bullet Description. As a construct of model operability, the Bullet model takes the bullet being ready as an input from the magazine, in order to make a calculation on the dud rate of a given bullet. This event triggers two outputs - a status of whether the current bullet in question is a dud or not, and it re-transmits the bulletReady assessment of the Magazine model.

Bullet Specification. The following is a formal specification of the Bullet, using the DEVS Formalism:

M_{Bullet} = <X, S, Y, δ_{int}, δ_{ext}, λ, ta>, where

X = {(in_bulletReady, {0,1})};

S = {state ∈ {passive, active}, preparationTime, isDud ∈ {0,1}, bulletRdy ∈ {0,1}}

Y = {(out_bulletReady, {0,1})};

```
δint (s) {
        passivate();
}

δext (s,e,x) {
        if (bullet is ready)
            set as a dud 5% of time        //can be approximated by another distribution
        state = active;
        ta (state) = preparationTime;
}

λ (s) {
        sendOutput(time, out_isDud, isDud)
        sendOutput(time, out_bulletReady, bulletRdy)
}

ta (state) = preparationTime;
```

4.2. Trigger Assembly Atomic Model

Trigger Assembly Description. This Trigger Assembly is a complex mechanism, which connects the trigger (in_triggerPressed), and firing selector (in_firingSelector), in order to make a determination as to whether the Sear, and in turn, the Bolt Assembly, can be released (out_releaseBolt). There are three firing modes: 1) Safe, which will not allow the bolt to be released, regardless of other inputs; 2) Single shot, which must have a trigger being pressed and released for each bullet to be shot; and 3) automatic, which requires just one constant trigger pressure to release a stream of bullets. As a construct of model operability, it takes as an input from the Chamber model whether the bolt has been pushed back (in_boltBack), and will cause another assessment (based on trigger pressure and firing mode) of whether to release the bolt for another firing cycle.

Trigger Assembly Specification. The following is a formal specification of the Trigger Assembly, using the DEVS Formalism:

$M_{TrigAssy}$ = <X, S, Y, δ_{int}, δ_{ext}, λ, ta>, where

X = {(in_triggerPressed, {0,1}); (in_firingSelector, {0,1,2}); (in_boltBack, {0,1})};

S = {state ∈ {passive, active}, preparationTime, FSstate ∈ {0,1,2}, triggerPull ∈ {0,1}}

Y = {(out_releaseBolt, {0,1})};

```
δint (s) {
        passivate();
}

δext (s,e,x) {
        if (port == in_triggerPressed)
            assign local variable triggerPull to this value
        else if (port == in_firingSelector)
            assign local Firing Selector state (FSstate) to this value
        state = active;
        ta (state) = preparationTime;
}

λ (s) {
        switch (FSstate)
            case 0:        //ie. the weapon is on safe, and shouldn't do anything
                    if trigger has been pulled, sendOutput(time, out_releaseBolt, do not
                    release bolt)
            case 1:        //ie. the weapon is on single shot
                    if trigger has been pulled, sendOutput(time, out_releaseBolt,
                    release bolt)
                    re-set triggerPull, requiring trigger to be pulled for next shot
            case 2:        //ie. the weapon is on automatic
                    if trigger has been pulled, sendOutput(time, out_releaseBolt,
                    release bolt) //will keep firing if the bolt comes back and still on
                    automatic

if bulletsLeft >= 0, and the magazine is seated, then a bullet is ready
        else bullet is not ready
        sendOutput(time, out_bulletReady, bulletReady)
}

ta (state) = preparationTime;
```

4.3. Bolt Assembly Atomic Model

Bolt Assembly Description. This Bolt Assembly interacts with all of the other sub-assemblies of the Rifle. It requires the MagAssy to notify it if a bullet is ready to be loaded (in_bulletReady), the TrigAssy notifies it if the bolt is held by the sear, and can be released or not (in_releaseBolt), and it is informed if it is being pulled / pushed back by either the external cocking handle or the Chamber (in_boltBack). It informs the Chamber as to whether a bullet has been successfully loaded or not (90% of the time it will), and it informs the external world as to its position, whether forward (0), back (1), or obstructed (2) by a blockage (which happens the other 10% of the time). Of note, these probabilities are pure estimate, and can be substituted by more accurate / representative probabilistic distributions, if required.

Bolt Assembly Specification. The following is a formal specification of the Bolt Assembly, using the DEVS Formalism:

$M_{BoltAssy} = \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$, where

X = {(in_bulletReady, {0,1}); (in_releaseBolt, {0,1}); (in_boltBack, {0,1})};

S = {state ∈ {passive, active}, preparationTime, boltFree ∈ {0,1}, readyBullet ∈ {0,1}, boltState ∈ {0,1,2}}

Y = {(out_boltPosn, {0,1,2}), (out_bulletLoaded, {0,1})};

δ_{int} (s) {

passivate();

}

δ_{ext} (s,e,x) {

if (port == in_bulletReady)
assign local variable readyBullet to this value
else if (port == in_releaseBolt)
if the bolt is released and the bolt is currently back (=1)
assign local variable boltFree to 1 as well
else if (port == in_boltBack) //ie. either by Chamber or cocking handle
if bolt is being sent back and it is forward, then set boltState = back
state = active;
ta (state) = preparationTime;
}

λ (s) {

if the bolt is free and it is back
if a bullet is ready

90% successfully load bullet in chamber, bring bolt forward,
sendOutput(time, out_bulletloaded, bullet is loaded)

10% of time get a jam/ obstruction (boltPosn = 2)
sendOutput(time, out_bulletloaded, bullet not loaded)

else bullet is not ready
bring bolt forward, but no bullet loaded in chamber
sendOutput(time, out_bulletloaded, bullet not loaded)

sendOutput(time, out_boltPosn, boltPosn)
}

ta (state) = preparationTime;

4.4. Chamber Atomic Model

Chamber Description. Inputs are received from the BoltAssy, as to whether a bullet is loaded or not. As a construct of model operability, the chamber finds out if a loaded bullet is a dud or not, from the MagAssy model. Based on its inputs, it attempts to fire the bullet, and outputs a bulletFired message, a brass shell casing message, and it notifies other sub-assemblies that the bolt has been pushed back, initiating another loading cycle.

Chamber Specification. The following is a formal specification of the Chamber, using the DEVS Formalism:

$M_{Chamber}$ = <X, S, Y, δ_{int}, δ_{ext}, λ, ta>, where

X = {(in_isDud, {0,1}); (in_releaseBolt, {0,1}); (in_boltBack, {0,1})};

S = {state ∈ {passive, active}, preparationTime, dudBullet ∈ {0,1}, bulletIn ∈ {0,1}}

Y = {(out_boltBack, {0,1}), (out_bulletFired, {0,1}), (out_casing, {0,1})};

```
δint (s) {

        passivate();

}

δext (s,e,x) {

        if (port == in_isDud)
            assign local variable dudBullet to this value
        else if (port == in_bulletLoaded)
            assign local variable bulletIn to this value
            state = active;
            ta (state) = preparationTime;        //only pass time if bullet is loaded
}

λ (s) {

        if it is a good bullet (non-dud) and there is a bullet loaded in the chamber
            sendOutput(time, out_boltBack, 1);
            sendOutput(time, out_bulletFired, 1);
            sendOutput(time, out_casing, 1);
            re-set local variables

}

ta (state) = preparationTime;
```

5. Test Cases and Execution Analysis

While the system was conceived from the “Top Down”, the testing was performed from the “Bottom Up”. As such, the following test cases will be presented from the basic Atomic Models, up to the complex Rifle Coupled Model.

Of note, the coding of this system in CD++ was based upon the “Cruise Control System” CD++ model, which served as a guide on CD++ coding style and coding techniques. Many thanks to the authors of that model.

5.1. Bullet Atomic Model

The Bullet atomic model is a simple model, that should output a dud bullet 5% of the time. A sample of 11 bullets were executed repeatedly, with various dud rate probabilities, and gave expected results. Below is sample INPUT/OUTPUT, where it happened that a dud bullet was encountered in the first instance. Of note, preparationTime was set to 0, to represent that this assessment should not expend time.

INPUT (Bullet.ev)	OUTPUT (Bullet.out)
00:00:00:100 in_bulletReady 1 00:00:00:200 in_bulletReady 1 00:00:00:300 in_bulletReady 1 00:00:00:400 in_bulletReady 1 00:00:00:500 in_bulletReady 1 00:00:00:600 in_bulletReady 1 00:00:00:700 in_bulletReady 1 00:00:00:800 in_bulletReady 1 00:00:00:900 in_bulletReady 1 00:00:01:000 in_bulletReady 1 00:00:01:100 in_bulletReady 1	00:00:00:100 out_isdud 1 00:00:00:200 out_isdud 0 00:00:00:300 out_isdud 0 00:00:00:400 out_isdud 0 00:00:00:500 out_isdud 0 00:00:00:600 out_isdud 0 00:00:00:700 out_isdud 0 00:00:00:800 out_isdud 0 00:00:00:900 out_isdud 0 00:00:01:000 out_isdud 0 00:00:01:100 out_isdud 0

Figure 5.1 - Bullet Testing

5.2. Magazine Atomic Model

The Magazine atomic model is given an initial seating, and bullet count. Various initial bullet counts were tested, along with sequential bullet loadings, to reduce the bullet count. The sample INPUT/OUTPUT below demonstrate how the model worked correctly. Of note, an initial bullet count of 10 bullets are shot and expended with 10 successive in_bulletLoaded messages, after which, out_bulletReady is outputted as 0, indicating that there are no more bullets in the magazine. In actuality (as shown later), there would not be any excess in_bulletLoaded messages which would generate out_bulletReady values of 0, as was deliberately done here. A poor magazine seating was tested as well, and this is shown as an input of in_initMagSeating of 0. This generates a out_bulletLoaded message of value 0, which is as expected.

INPUT (Magazine.ev)	OUTPUT (Magazine.out)
00:00:00:100 in_initMagSeating 1	00:00:00:100 out_bulletready 1
00:00:00:100 in_initBullets 10	00:00:00:300 out_bulletready 1
00:00:00:300 in_bulletLoaded 1	00:00:00:400 out_bulletready 1
00:00:00:400 in_bulletLoaded 1	00:00:00:500 out_bulletready 1
00:00:00:500 in_bulletLoaded 1	00:00:00:600 out_bulletready 1
00:00:00:600 in_bulletLoaded 1	00:00:00:700 out_bulletready 1
00:00:00:700 in_bulletLoaded 1	00:00:00:800 out_bulletready 1
00:00:00:800 in_bulletLoaded 1	00:00:00:900 out_bulletready 1
00:00:00:900 in_bulletLoaded 1	00:00:01:000 out_bulletready 1
00:00:01:000 in_bulletLoaded 1	00:00:01:100 out_bulletready 1
00:00:01:100 in_bulletLoaded 1	00:00:01:300 out_bulletready 1
00:00:01:300 in_bulletLoaded 1	00:00:01:400 out_bulletready 0
00:00:01:400 in_bulletLoaded 1	00:00:01:500 out_bulletready 0
00:00:01:500 in_bulletLoaded 1	00:00:01:600 out_bulletready 0
00:00:01:600 in_initMagSeating 0	00:00:01:800 out_bulletready 0
00:00:01:600 in_initBullets 10	00:00:01:900 out_bulletready 0
00:00:01:800 in_bulletLoaded 1	00:00:02:000 out_bulletready 0
00:00:01:900 in_bulletLoaded 1	
00:00:02:000 in_bulletLoaded 1	

Figure 5.2 - Magazine Testing

5.3. Magazine Assembly Coupled Model

The Magazine Coupled Model, encompassing the Magazine and Bullet atomic models, is a static entity, and combines their outputs. As it is a static entity, preparationTime was set to 0 for both atomic models.

The model worked as expected, and was tested for proper / improper magazine seating, various in_initBullets values, and it demonstrated expected out_isDud and out_bulletReady values.

The sample INPUT/OUTPUT below demonstrate how out_isDud and out_bulletReady messages are generated successfully, and after 10 bullets are expended, no more messages are sent out.

INPUT (MagAssy.ev)	OUTPUT (MagAssy.out)
00:00:00:100 in_initMagSeating 1	00:00:00:100 out_isdud 0
00:00:00:100 in_initBullets 10	00:00:00:100 out_bulletready 1
00:00:00:200 in_bulletLoaded 1	00:00:00:200 out_isdud 0
00:00:00:300 in_bulletLoaded 1	00:00:00:200 out_bulletready 1
00:00:00:400 in_bulletLoaded 1	00:00:00:300 out_isdud 0
00:00:00:500 in_bulletLoaded 1	00:00:00:300 out_bulletready 1
00:00:00:600 in_bulletLoaded 1	00:00:00:400 out_isdud 0
00:00:00:700 in_bulletLoaded 1	00:00:00:400 out_bulletready 1
00:00:00:800 in_bulletLoaded 1	00:00:00:500 out_isdud 0
00:00:00:900 in_bulletLoaded 1	00:00:00:500 out_bulletready 1
00:00:01:000 in_bulletLoaded 1	00:00:00:600 out_isdud 0
00:00:01:100 in_bulletLoaded 1	00:00:00:600 out_bulletready 1
00:00:01:200 in_bulletLoaded 1	00:00:00:700 out_isdud 0
00:00:01:300 in_bulletLoaded 1	00:00:00:700 out_bulletready 1
00:00:01:400 in_bulletLoaded 1	00:00:00:800 out_isdud 0
00:00:01:500 in_bulletLoaded 1	00:00:00:800 out_bulletready 1
00:00:01:600 in_bulletLoaded 1	00:00:00:900 out_isdud 0
00:00:01:700 in_bulletLoaded 1	00:00:00:900 out_bulletready 1
	00:00:01:000 out_isdud 0
	00:00:01:000 out_bulletready 1
	00:00:01:100 out_isdud 0
	00:00:01:100 out_bulletready 1

Figure 5.3 - Magazine Assembly Testing

5.4. Trigger Assembly Atomic Model

The Trigger Assembly model was tested through all the firing modes, and possible inputs, and demonstrated expected results.

Below is sample INPUT / OUTPUT. Of note, initially, the TrigAssy is set to Safe (in_firingSelector 0), and the trigger is pressed. As expected, nothing happens. The sample INPUT shows the trigger being released by in_triggerPressed 0.

The system is set to Single shot mode (in_firingSelector 1), the trigger is pressed, and expectedly, after repeated in_boltBack messages, still only the initial out_releaseBolt 1 message is sent, showing that the trigger must be pressed again before the bolt will be released again in Single shot mode.

The system is set to automatic mode, and correctly, with the initial trigger pressure, and every subsequent in_boltBack 1 message, the bolt is released every time, until the trigger pressure stops (in_triggerPressed 0).

A default time advance of 00:00:00:050 was set, to represent the releasing of the bolt action.

INPUT (TrigAssy.ev)	OUTPUT (TrigAssy.out)
00:00:00:100 in_firingSelector 0 00:00:00:200 in_triggerPressed 1 00:00:00:300 in_triggerPressed 0 00:00:01:000 in_firingSelector 1 00:00:01:100 in_triggerPressed 1 00:00:02:000 in_boltBack 1 00:00:02:100 in_boltBack 1 00:00:02:200 in_boltBack 1 00:00:02:300 in_triggerPressed 0 00:00:03:000 in_firingSelector 2 00:00:03:100 in_triggerPressed 1 00:00:04:000 in_boltBack 1 00:00:04:100 in_boltBack 1 00:00:04:200 in_boltBack 1 00:00:04:300 in_boltBack 1 00:00:04:400 in_boltBack 1 00:00:04:500 in_boltBack 1 00:00:04:600 in_boltBack 1 00:00:04:700 in_triggerPressed 0	00:00:00:250 out_releasebolt 0 00:00:01:150 out_releasebolt 1 00:00:03:150 out_releasebolt 1 00:00:04:050 out_releasebolt 1 00:00:04:150 out_releasebolt 1 00:00:04:250 out_releasebolt 1 00:00:04:350 out_releasebolt 1 00:00:04:450 out_releasebolt 1 00:00:04:550 out_releasebolt 1 00:00:04:650 out_releasebolt 1

Figure 5.4 - Trigger Assembly Testing

5.5. Bolt Assembly Atomic Model

The sample INPUT/OUTPUT below demonstrate that the Bolt is only successful in loading a bullet when a bullet is ready (in_bulletReady 1), the bolt is pulled/pushed back initially (in_boltBack 1), AND is allowed to be released by the TrigAssy (in_releaseBolt 1). This is shown initially at 00:00:00:320 in the output. However, erroneous successive in_boltBack and in_bulletReady messages do not result in any action from the bolt, until the bolt is released, after it being back with a bullet ready. Various errors were also sent, and correctly did not result in any bullets being loaded. Of note, this particular sample OUTPUT documents a chance bolt obstruction (out_boltPosn 2 - set to occur 10% of the time in this execution), which is representative of the expected behaviour, showing that also a bullet was not loaded (out_bulletLoaded 0). A preparationTime = 00:00:00:020 was selected to represent the rapid movement of the bolt.

INPUT (BoltAssy.ev)	OUTPUT (BoltAssy.out)
00:00:00:100 in_bulletReady 1 00:00:00:200 in_boltBack 1 00:00:00:300 in_releaseBolt 1 00:00:00:400 in_boltBack 1 00:00:00:500 in_boltBack 1 00:00:00:600 in_boltBack 1 00:00:00:700 in_bulletReady 1 00:00:00:800 in_bulletReady 1 00:00:00:900 in_bulletReady 1 00:00:01:000 in_releaseBolt 1 00:00:01:100 in_releaseBolt 1 00:00:01:200 in_releaseBolt 1 00:00:01:300 in_bulletReady 1 00:00:01:400 in_boltBack 1 00:00:01:500 in_releaseBolt 1 00:00:01:600 in_bulletReady 1 00:00:01:700 in_boltBack 1 00:00:01:800 in_releaseBolt 1 00:00:01:900 in_bulletReady 1 00:00:02:000 in_boltBack 1 00:00:02:100 in_releaseBolt 1 00:00:02:200 in_bulletReady 0 00:00:02:300 in_boltBack 1 00:00:02:400 in_releaseBolt 1 00:00:02:500 in_bulletReady 1 00:00:02:600 in_boltBack 1 00:00:02:700 in_releaseBolt 0	00:00:00:320 out_bulletloaded 1 00:00:00:320 out_boltposn 0 00:00:01:020 out_bulletloaded 1 00:00:01:020 out_boltposn 0 00:00:01:520 out_bulletloaded 1 00:00:01:520 out_boltposn 0 00:00:01:820 out_bulletloaded 1 00:00:01:820 out_boltposn 0 00:00:02:120 out_bulletloaded 0 00:00:02:120 out_boltposn 2

Figure 5.5 - Bolt Assembly Testing

5.6. Chamber Atomic Model

The Chamber model was tested with all possible inputs. The sample INPUT/OUTPUT below demonstrates how the model does not react / output fired bullets in response to erroneous in_isDud messages, nor in_bulletLoaded messages. It only fires a bullet, correctly outputting out_boltBack, out_bulletFired and out_casing values at times 00:00:00:220 and 00:00:00:620, after there is a non-dud bullet AND a bullet loaded present.

INPUT (Chamber.ev)	OUTPUT (Chamber.out)
00:00:00:100 in_isDud 0 00:00:00:200 in_bulletLoaded 1 00:00:00:300 in_isDud 0 00:00:00:400 in_isDud 0 00:00:00:500 in_isDud 0 00:00:00:600 in_bulletLoaded 1 00:00:00:700 in_bulletLoaded 1 00:00:00:800 in_bulletLoaded 1 00:00:00:900 in_isDud 1 00:00:01:000 in_bulletLoaded 1 00:00:01:100 in_isDud 0 00:00:01:200 in_bulletLoaded 0 00:00:01:300 in_bulletLoaded 0 00:00:01:400 in_bulletLoaded 0 00:00:01:500 in_bulletLoaded 0	00:00:00:220 out_boltback 1 00:00:00:220 out_bulletfired 1 00:00:00:220 out_casing 1 00:00:00:620 out_boltback 1 00:00:00:620 out_bulletfired 1 00:00:00:620 out_casing 1

Figure 5.6 - Chamber Testing

5.7. Rifle Coupled Model

The complete Rifle Coupled Model was tested extensively with all the possible values of inputs of its component sub-models. The testing scheme aggregated the sub-components testing schema, including various inInitMagSeating and inInitBullet values, erroneous inTriggerPressed inputs (0 or 1), inFiringSelector inputs (0,1 or 2), and inHandlePosn inputs (0 or 1).

The model responded as expected. Of note, in the sample INPUT/OUTPUT below, the model correctly fires when in automatic mode after time 00:00:00:480. This particular sample OUTPUT was chosen to show the weapon jam at time 00:00:00:620, which causes all firing to stop.

The timing represented here is an aggregate of the times taken for the BoltAssy, the Chamber and TrigAssy.

INPUT (Magazine.ev)	OUTPUT (Magazine.out)
00:00:00:100 inInitMagSeating 1 00:00:00:100 inInitBullets 10 00:00:00:200 inFiringSelector 2 00:00:00:300 inHandlePosn 1 00:00:00:400 inTriggerPressed 1 00:00:00:800 inTriggerPressed 0 00:00:02:100 inInitMagSeating 1 00:00:02:100 inInitBullets 30 00:00:02:200 inFiringSelector 0 00:00:02:300 inHandlePosn 1 00:00:02:400 inTriggerPressed 1 00:00:04:100 inInitMagSeating 1 00:00:04:100 inInitBullets 10 00:00:04:200 inFiringSelector 1 00:00:04:300 inHandlePosn 1 00:00:04:400 inTriggerPressed 1 00:00:04:500 inTriggerPressed 1 00:00:04:600 inTriggerPressed 1 00:00:04:700 inTriggerPressed 1 00:00:04:800 inTriggerPressed 1 00:00:04:900 inTriggerPressed 1 00:00:05:000 inTriggerPressed 1 00:00:05:100 inTriggerPressed 1 00:00:05:200 inTriggerPressed 1 00:00:05:300 inTriggerPressed 1 00:00:05:400 inTriggerPressed 1 00:00:05:500 inTriggerPressed 1 00:00:05:600 inTriggerPressed 1 00:00:05:700 inTriggerPressed 1	00:00:00:460 outboltposn 0 00:00:00:480 outbulletfired 1 00:00:00:480 outcasing 1 00:00:00:540 outboltposn 0 00:00:00:560 outbulletfired 1 00:00:00:560 outcasing 1 00:00:00:620 outboltposn 2

Figure 5.7 - Rifle Couple Model Testing

6. References

The modelling of this weapon is based on Open-Source documentation and references, independent of any detailed military or manufacturer specifications. Pictures of the weapon are available at the references. The references are:

Colt Canada, 2009. C7 Rifle. Colt Canada Website. http://www.coltcanada.com/c7-page.htm. Accessed: Nov 2, 2009.

Harris, T., 2001. How Machine Guns Work. In HowStuffWorks.com. http://science.howstuffworks.com/machine-gun.htm. Accessed: Nov 2, 2009.

Janes Infantry Weapons, 2009. Colt Canada C7 5.56mm Assault Rifle. In Janes Online. http://www.janesonline.com. Accessed: Nov 2, 2009.