Implementation of Automated Combat Resolution Using a Reduced Set of Rules from Abstract Combat System, Alpha Strike and Battleforce

Luana C. Coppio

MegaMek / MekHQ / MegaMekLab

January 23th of 2025

I would like to thank the current team of volunteers, developers, testers, and lore keepers. Your hard work and countless hours of effort have made this project possible. It exists because of your dedication to something we all value deeply.

Abstract

While MegaMek Total Warfare combat is engaging, certain situations require streamlined combat resolutions to efficiently advance the overarching narrative. Relying on the Princess AI can alleviate the need for manual play, but it is computationally intensive and slow, with full scenarios taking several minutes to conclude. Since time is a valuable resource, an automated combat resolution method was developed. Based on a subset of the Abstract Combat System rules, this method—named Abstract Combat Auto Resolution (ACAR)—employs specialized heuristic algorithms that heavily depend on the force units and pilot experience. Initial findings suggest that this method offers a promising balance between narrative progression and the preservation of strategic depth.

Keywords: auto-resolve, ACS, SBF, MegaMek, MekHQ, bot, Princess, AI, ACAR

Implementation of automated combat resolution using reduced set of rules from Abstract Combat System, Alpha Strike and Battleforce

Automatic combat resolution must be fast. However, the common technique of relying on Princess for this task is far from efficient. To address this, we set out to create a method capable of reducing large battles—spanning dozens, hundreds, or even thousands of units—into a process that can be simulated quickly and reliably while accounting for the decisions made in MekHQ.

The *Interstellar Operations BattleForce 3rd Edition* includes rules for what it describes on page 234 as follows:

"The Abstract Combat System (ACS) allows players to fight large, multi-regiment battles without the need to track every detail of the forces involved. Where Total Warfare sees players fighting for a small scrap of land or single building objectives, ACS allows players to play out the conflict for entire worlds."

This system aligns well with our objectives: a method to represent single scenarios where two or more armed forces face off, while focusing only on a single combat encounter. However, much of the original ACS rule set is unnecessary for our purpose, as it includes elements like weeks-long attrition, movement over vast distances, supply line management, and similar complexities.

The ACAR system was developed specifically for single-formation battles. This required removing rules tied to movement across a board, adopting “less bookkeeping rules,” and introducing a more granular approach to unit damage during combat. Additionally, some rules were adapted to better suit the intent of our simulation. Instead of outright throwing away the game board, ACAR implements a one-dimensional board, where the units can move forwards and backward, allowing them to tail targets, stay away from enemies, etc, this proved to be a

very important implementation since it allows for a better feeling of correctness - units would take multiple turns to meet the first enemy, faster units can fire sooner, etc.

Lastly, it was essential to ensure that ACAR could generate a valid post-game event compatible with MekHQ's Scenario Resolution Wizard.

Initial Topic Research

Simon Parkin describes in his book *A Game of Birds and Wolves* the creation of a wargame during world war II, by the retired captain Gilbert Howland Roberts and Women's Royal Naval Service which would allow them to quickly iterate new strategies to defend merchant convoys against u-boat attacks. The ability to safely and quickly simulate hundreds of battles per week allowed other captains to devise and test defense strategies, and ultimately doomed the u-boat effectiveness in the sea.

Many grand strategy games have auto-resolve features in it, to allow the player to focus on what really matters at the time, and not all battles are turning points or critical in a campaign. Before arriving at the current configuration of the auto-resolve, many other techniques were explored. The prerequisites are: Units need to be damaged and/or destroyed, individual units have to matter, the end result of the simulation needs to "feel" like the end result of a normal MegaMek combat.

EDNA

EDNA is a system that allows for strategic movement without the need for unit removal, it is very simple and abstract things like morale, supply lines, combat strain and stress. It however was too abstract and would not translate well for a combat resolution which requires units to be destroyed. Its rules are interesting but there was no way to implement it for evaluating results without a lot of additions.

SCRUD

Simple Combat Resolution Using Dice is a nice tool, but it expects units to be equivalent, some can use small modifiers, so some modifiers were added like +1 for meks, -2 for infantry, etc. It did work in a way that is interesting, but it also abstracted too much from the units, alienating the force from the result of the game. We implemented a version of SCRUD where BV would give one dice to roll for every 50 BV rounded up, and instead of removing units it removes BVs (50 BV for each dice), a “destruction budget” accumulates in each round and units that fit that budget are destroyed, if there is budget left it carries over to the next round. The system was incredibly simple, but resulted in end games that didn’t feel natural, with complete annihilation of one force while the other had almost no damage.

MegaMek Headless

This was an idea of running a scenario in a headless format, without graphics, just a loading screen and report on the actions streaming for the user, but this would not considerably increase the speed of the resolution, so it was discarded.

Alpha Strike (AS)

Simplified units, but too complex game play. Abstracting movement was also hard to do, implementing all rules necessary to simulate would take too long and would also make it too susceptible to bugs given the amount of code necessary to implement it all. However some elements of it are reserved to be explored later if they fit, like the simplification of the units.

Strategic Battleforce (SBF) and Abstract Combat System (ACS)

More abstracted rule system allows for battles of a larger scale, it looks like it would be a good fit for the implementation of the auto-resolution, the aerospace “mode” isn’t necessary for

our implementation. It still has a somewhat complex movement system, but it seems to be easy to just remove this part.

ACS is an even more abstract rule system, with even simpler rules, focused in world-invasion scale campaigns, uses the same setup by SBF, but makes things faster. The reduced number of rules, decisions and effects makes it an even better fit. We can strip the rules that add things we don't, since this auto-resolve is focused on a single skirmish we don't need to use things like movement, *aerospace phase* and *scouting and detection phase*, the game rules still has reference to each individual unit making them important, and reduces the number of rolls per turn to one per formation, making allowing us to consider that each team is a single thing for most of the considerations.

Method

Abstract Combat Auto-Resolution

ACAR has 3 main stages: Force Setup, Simulation, and Post Combat Resolution.

The current implementation is limited in what types of entities it can convert to Alpha Strike, currently it cant do ArmlessMek, EjectedCrew, FighterSquadron, EscapePod and TeleMissile.

The auto resolution has three different “force to formation” setups, which will be discussed in its own topic.

Data Sources

Aside from the type of unit restriction, ACAR uses the complete unit with its crew for its simulation, the variant, tonnage, speed, jump capability, the pilot piloting and gunnery skill contributes to the unit “skill”, which is also a source for its “tactic” and “morale”.

ACAR requires that units to be “abstracted” into smaller and less complex things, so it is first converted into an Alpha Strike element, then one or more, up to four, elements are converted into an Strategic Battleforce (SBF) unit, which is in turn divided in groups of one up to 4 SBF units, forming one ACS/SBF Formation.

For the purposes of ACAR, the grouping of units is mostly useful for how turns are taken, number of actions each team takes, and for rolling all non-attack actions. They mostly mean the average value of skill, morale and tactics for your entire team is used instead of individual units having to roll for each individual action. ACAR reduces most things that don't “cause damage” to a single roll.

Currently the planetary conditions do not interfere in the combat resolution. Neither does the type of board, type of objective, or unit role or unit abilities. Those are possible future iterations. The size of the board does have effects, as the board is represented as a “ruler” which the size is the diagonal of the board, so a 20x20 board is represented as a ruler of 28 marks.

Algorithm Design

Force Setup. The forces can be transformed into formations in one of the three following ways:

- Singleton Force: Every unit of each force is represented as an individual unit in ACAR, this has the benefit of allowing a more granular processing of the events. This is ideal for small force combats, like a one lance vs one lance.
- Flatten Forces: Each “lowest level” of your forces (typically the lance) is elevated to the level of a force, if you have 4 lances, they become 4 forces.
- Use Current Forces: Keep the player forces as is.

The forces then are converted into formations, the way each is going to be interpreted into a formation depends on some factors:

- Entity as formation: Every unit is treated as a single formation each.
- Entity as a unit: Every lance will be turned into a formation each, and the units it has will become a “SBF Unit” inside of it. An SBF Unit represent a number of AS elements (alpha strike elements, like two meks, or a couple of vehicles). In this mode each lance is a formation, meaning all the units will move together and will shoot at the same target formation.
- Lowest force as unit: The lance itself will be considered a unit, meaning that a single company can be represented as a formation with 4 units, each unit is an entire lance. This allows for a more strategic simulation, but any unit being destroyed means a whole lance being wiped out.

Simulation. The auto-resolve implements a reduced and adapted set of rules of ACS, most notably the without the movement rules, instead a unidimensional board (a ruler) is used to represent the separation between the formations. Each formation will then rely on a simple AI which is based on the unit role to decide if they need to find an enemy, take cover, change targets, etc.

Simplifications

Because ACAR sits in a strange space, it focuses in what effectively can be described as a small skirmish, which is the focus of Megamek and the TW ruleset, however TW is too complex to be reliably simulated without alot of both computing power and development time, so a layer of abstraction must be added, there comes Alpha Strike, which in itself also has is very complex, with alot of focus on terrain, distances, speeds, abilities. So another layer of abstraction that

already exists and is successful is SBF, which does implement combat as a very simple and abstract thing, But still, their scale is another, it works for large forces, combats that span days over many kilometers of ground. And it still has a lot of very complex movement rules. One extra layer of abstraction and we move to ACS, which basically can be used to represent full scale planetary invasions. This abstracts combat to a point where it is almost just a dice roll, but it still takes into consideration unit skill and all its particularities.

Reducing the number or rules and mechanics has a benefit, if the abstraction is "good enough" the results are materially indistinguishable from an ACAR simulation and a game of Alpha Strike, and that's the objective I tried to reach with this implementation.

Implementation

As with the rest, this is implemented in Java, entirely inside MegaMek as an independent module.

Test scenarios were implemented with JUnit and mockito to check a high number of combat simulations (1000 runs take something around 5 minutes in a Macbook Pro M2) to evaluate the system fairness.

The Mechanics

Inputs

The system relies only on one input - the forces. The system will create mirror units from the original units from each force, create a conversion to Alpha Strike copy for each of them with the same ID so they can be referenced one to another later, then transform the AS elements into SBF units and the SBF units into ACS formations. The player can't really do anything other than stack the odds in their favor by loading their best pilots in their best meks and hoping for the best.

Process

The auto-resolve combats follow a very simple structure:

- Starting Scenario Phase;
- Initiative Phase;
- Deployment Phase;
- Movement Phase;
- Firing Phase;
- End Phase;
- Victory Phase;

Each implement as much as necessary of the original rules on how things resolve.

Starting Scenario Phase. Runs only once. It sets up initiative, resets units, creates the board representation and sets up deployment tables in memory.

Initiative Phase. This runs every round, it rolls the initiative for each individual formation, and increments the round counter.

Deployment Phase. If there are units to deploy in the current round they are deployed now, if not, moves to the next phase.

Movement Phase. Each formation tries to move towards a target enemy, if there are no enemies close to them they will instead move towards the middle of the board. The formation may also decide to move through partial cover if it is available and if they are exposed to enemy fire. Formations may also move towards or away from a target to keep them in their preferred attack distance, or they may move towards the closest edge of the board to withdraw.

Firing Phase. A formation can shoot against another formation with every single (SBF Unit) it has, if the formation has two or more units it can target between one and two targets as formations.

During the resolution, the combat rolls are done, and damage is applied from individual (alpha strike elements) and applied at random (individual alpha strike) elements of the target formation. The amount of damage is marked in the elements, then the total damage and armor is "pushed up" and consolidated in the (SBF) unit. (This is exclusive to ACAR).

It is also checked for critical damage, if the unit suffered from high stress or not (this one is using SBF rules instead of ACS rules since the combat is of a much smaller scale).

End Phase. First thing in the end phase is to check for unit destruction. If any AS element dies, (has 0 armor and 0 structure, or was destroyed with a lucky critical strike), they are removed from the unit. If the unit dies, the unit is removed from the formation, if there are no units left in the formation, the formation is removed from the simulation. All original "entities" (the original Megamek units that we all know and love) are then marked with the "removal condition" according to the reason that its alpha strike mirror was destroyed, which can be devastated, destroyed by ejection, or destroyed but left in a salvageable state.

Then it will check for decrease in morale, any formation that had a deeply emotional episode has to test if they won't lose morale, unless they are already routed.

After that, every formation that is either of routed morale or crippled will attempt to withdraw from the scenario, unless they are the last remaining team, in which case they will say that was their plan all along.

For last there is the nerve recovery event, where each formation with morale lower than normal will try to regain a little bit of it.

If there are still two or more teams in the simulation, it moves to the initiative phase, otherwise it moves to the victory phase.

Victory Phase. This phase will consolidate the results of the game, every unit that was destroyed will have its original entity receive damage equivalent to the type of destruction it suffered. A devastated unit will be ground down to a pulp, one that is salvageable will be beater alot, one ejected will be beaten alot and suffer the ejection damage, any unit that somehow survived (retreated units or those left at the end of a successful combat for example) will take an amount of damage that is equivalent to the percentage of damage the alpha strike mirror unit suffered. The pilot will suffer damage accordingly. This is a simplified damage model so there are no ammo explosions decorrent of this damage application, pilot death due to life-support destruction or cockpit destruction. And pilots can't die if the unit survived at the end of the combat.

Outputs

At the end of an auto-resolved scenario, you are greeted with a simple dialog and question - You either lost, won or drew the scenario, do you want to declare that you have control over the battlefield after the end of it? - This is something that the normal combat resolution also does, and the reason is simple, we haven't programmed a way to determine if you should control the battlefield at the end of the scenario before we run down the objectives, and it may depend on the narrative too.

The auto-resolution also gives a standard post-combat resolution event, which allow us to go through with a post-combat wizard, so we can pick the salvage, prisoners, cred all uncredited kills, and check the objectives. If you cancel this screen the resolution will stop and cancel the

result you just had, which does allow you to run the auto-resolve as many times as you want with no repercussions.

Measures

Assessments and Measures

The implementation was tested against 4 different scenarios built with JUnit and Mockito, each one run 1000 times and their results logged.

Scenario 1 (S1 balanced). Both teams are exactly the same, same meks, same crew skill and experience. Expected victory distribution to be split 50/50, and also expects the number of draws to be high (more than 10% but less than 20%) because to achieve that, either both units destroy each other or both have to withdraw from the board.

Scenario 2(S2 unbalanced). Same overall makeup as S1, except that in this one the Team 1 has one extra light mek with a green pilot. This mek added a small amount to the adjusted BV of the team 1 and should reflect in the results of the tests.

Scenario 3 (S3 BV balanced). Both teams have different meks and pilot skill, but totals the same amount of BVs (6k adjusted BV). We expect to see differences in the victories distribution as different individual meks and pilot skill may influence the results of combats.

Scenario 4 (S4 BV/Skill balanced). Both teams have the same number of meks and pilots with the exact same skill level, and the adjusted BV is the same for both teams.

We expect to see differences in the victory distribution as the different meks offer different combat ranges and armor.

Results

Scenario resolution table

Scenario	Team 1 Victories	Team 2 Victories	Draws	Total Runs
S1 balanced	433	436	121	1000
S2 unbalanced	520	353	127	1000
S3 BV balanced	602	268	130	1000
S4 BV/skill Balanced	537	371	92	1000

The results align with expectations, which is a welcoming sign, however it doesnt take into consideration that the BV cost doesnt properly translate to PV cost which is very different when working with formations.

Benchmark

The simulations usually run under 300ms on a Macbook Pro M2, with 16GB of RAM, with 8GB RAM allocated for the JVM. The simulations inside MekHQ are run in parallel, one thread per cpu core, and 1000 simulations take an average of 25 seconds.

Outcomes

Implications

Tests with volunteers suggest that ACAR improves the game pacing, allowing for players to resolve scenarios that they would otherwise ignore or “start and flee” immediately. We also observed that the inherent variance and randomness of it will often put players under a pressure that they wouldn't normally have, with multiple units requiring repairs, destroyed and pilots hurt, be it because the player can be more careful in deciding which units they want to protect better when they control them, or because they are playing many more combats than they would otherwise.

ACAR also enables bigger companies, making it manageable with the very high number of combat scenarios Stratcon throws against companies with large TO&E's. Maneuvering large forces to combat also becomes a task that is way easier and can be resolved almost instantly with it.

Future Improvements

As discussed before, many features are missing, and some may see the light of day sooner than later, simulating combat on MegaMek instead of being limited to MekHQ, taking planetary conditions into consideration, those are just two of the future improvements for ACAR.

If you have any request about the abstract combat auto-resolution, please, leave a request for feature enhancement (RFE) on our github, or come talk to us on our discord!

References

Catalyst Game Labs (2024). Battletech Interstellar Operations Battleforce 3rd ed, 160-174, 234-248, 256-268.

Parkin, S. (2021). *Designing the Secret Game that Helped Win World War II*. Youtube. Retrieved December 25, 2024, from https://www.youtube.com/watch?v=01UiEi6HkEY

MapSymbs, Muat, T. (n.d.). *Matrix Games Download*. Retrieved December 25, 2024, from http://www.mapsymbs.com/wdmatrix1.html

MurdoK (2013, February). *EDNA and SCRUD*. *Wargaming Miscellany*. Retrieved December 25, 2024, from http://wargamingmiscellany.blogspot.com/2020/01/edna-and-scrud.html

MurdoK (2013). Scrud - Simple Combat Resolution Using Dice,. *MurdoK's MarauderS*, Retrieved December 25, 2024, from, http://murdocksmarauders.blogspot.com/2013/02/scrud-simple-combat-resolution-using.html