## <u>Question 1a – Project Proposal</u>

I initially started a project with similar goals as this one in 2001 and had the working title of MARS, this project was never completed. The renewed project started for the TM420 course is now under the title MARS4STARS.

If I where to restart this project again from scratch, there is not much I would change from my approach. However I have learned, at least in retrospect, of several serious problems in my approach to the initial MARS project.

One of the biggest stumbling blocks was the lack of prior design work. I had created no UML or other diagrams to help organize my work or to give me an easy overview of the code. I was working on the principal designing a section at the same time I was coding it.

With the current project I have resisted the temptation to cut short the UML design and start coding. I initially created a Class Diagram to represent all the data structures I thought I would need, as was the case with the initial MARS project. I did this because I already had a clear idea of the data structures.

However this time round I focused on several additional items, the first was on designing class relationships, without overly focusing on their internal data structures. This has lead to a far greater number of smaller classes than the few large classes in the initial MARS project.

Secondly, one of the possible future extensions planned for this project is to fully clone the entire game Stars with additional extensions. This has lead me to focus on a design that is fairly modular and easily extendable. I did not have this secondary goal in initial MARS project, which is another reason it ended up with fairly monolithic classes.

Thirdly, I started to divide up classes into several "types" depending on how there instances would be used during the lifespan of the program execution.
- "universe/rule-set data" - classes which would not be modified after initialization
- "user data" classes that the user would create/modify through the user interface
- "simulation data" classes used for calculating and storing the result/progress of a battle

In the initial MARS project, while I did make some distinction between "universe/rule-set data" and "user data", I did not make a similar distinction for "simulation data" and instead I included it with "user data". During the battle simulation, the "user data" needs to stay static and independent of the results of the "simulation data". Again this lead to large monolithic classes and a data structure that was not as "clean" as my current design.

After getting my thoughts down on the Class Diagram, I turned my focus on the Activity Diagrams. The process of working out all the details helped pinpoint the inconsistencies and missing classes within my Class Diagram thus allowing me to refactor it accordingly. Additionally the Activity Diagrams led me to realize the need for a stricter separation between "user data" and "simulation data".

Before starting the Activity Diagrams, I had the mistaken belief that I had the entire Class Diagram worked out perfectly, which is an assumption I also held after completing the data structures within the initial MARS project. It actually surprised me to realize that I had a few missing classes and needed several instances of refactoring the design. However whist working with the high level UML Class Diagram, it was fairly easy to do this refactoring once I had pinpointed what had been done. The efforts needed to refactor the code in the initial MARS project would have been significantly more, as the "design as you code" method requires altering all the code previously written that was affected the refactored section. Also without a high level overview, spotting these high level design problems by tracing through the code would have been a lot harder.

Again resisting the temptation to start coding, I started focusing on doing collaboration diagrams. This immediately reminded me that while I has a list of functions specified in the Activity Diagrams, I had not specified in which classes these functions would be placed. In doing the collaboration diagrams it would force me to explicitly specify all the functions that a class would need (in addition to its data members). This turned my attention back to the Class Diagram and with my mental model for how the collaboration diagrams would work, I flushed out the full specification of the classes. As before, the process of explicit specification showed up a few inconsistencies and other areas that need to be refactored.

This was not something I had done in the initial MARS project. Instead I focused my coding to align with the order of execution within the program (thus starting with the user interface and the data initialization functions). I would focus on coding a function and working out its dependencies on other class functions as I went.

The major benefit of going through the design process with UML diagrams, as shown above, has been that by explicitly describing each element of the system combined with the diagrammatic notation helps show up the inconsistencies and holes within a design and allows for each refactoring at the high level design. Additionally as the design is implementation free, refactoring becomes a lot simpler as there is no dependent code.

Another key mistake that I rectified was making the design far more philosophically object-orientated with a greater number of smaller classes. It also started me focusing on a function in terms of a black box with a specification without having to draw my attention onto its exact implementation.

When I set out to start the new MARS4STARS project, I did not have any these points in mind. I resisted the temptation to start coding immediately mainly due to the fact that I considered that the creation of UML diagrams was a requirement of the course. It is mainly in retrospect, by comparing the two projects, that I can see the mistakes I made in the initial MARS project.

**<u>Question 1b (i) – Literature Search</u>**

Targeting Order in Battles by Art Lathrop
http://www.starsfaq.com/articles/sru/art201.htm

**PROMPT criteria:**
**Presentation:**
The article was originally written as an ASCII newsgroup post and converted to HTML for web purposes with the line wrapping removed.

The article contains mathematical formulas written as pseudo-programming statements and uses commonly used characters for mathematical operators (such as * for multiplication and ^ for raise to the power of). There is no use of calculus or other advanced mathematical concepts used in the formulas.

Mathematical formulas are placed on new lines with a description above them. There is a blank line kept between formula sections and paragraphs of descriptive text.

It is fairly easy to read. The article was written for players of the game stars and is written in the context of the game. The article does not require any specialist knowledge to make sense of beyond basic maths.

**Relevance:**
The information matches my needs exactly. It provides an exact mathematical formula for the targeting algorithm used within a stars battle. It can be used directly within my program as part of of the functions that deal with choosing which ships should be targeted next in a battle.

**Objectivity:**
The article focuses on a concrete and testable mathematical formula rather than personal opinions. He has no vested interest in misrepresenting the data and it is likely that his formulas would have tested and publicly commented on by other members of the stars community.

**Method:**
While not explicitly mentioned, the likely process used for reverse-engineering the formula was through creating testbeds and observing the results of battles within the game and refining the formula accordingly. This is the standard method used for reverse-engineering the formulas used within the game.

**Provenance:**
The author, Art Lathrop is a long standing and respected member of the stars community. I know him through the rec.games.computer.stars newsgroup and I personally consider him to be a reliable source. He mentions ideas and articles from several other people in helping him come up with his formula.

While not explicitly mentioned on the website, I know that it was originally posted as ASCII text to the r.g.c.s. (rec.games.computer.stars) newsgroup. It was later placed as HTML onto Art Lathrop's personal website and additionally on the Stars-R-Us website (which is now offline). Later a copy of the article database section of the Stars-R-Us website (including this article) was put on the StarsFAQ website (by myself).

The article was posted to the rec.games.computer.stars newsgroup and subject the scrutiny of the stars community. I am unaware of anyone contesting the validity of this formula or proposing an alternative formula. Additionally by finding the article in the google.com newsgroup archive, it shows there was a follow up post by Bill Butler (another well respected member of the stars community) confirming the validity of the formula in his own tests.

**Timeliness:**
The article was originally written on 21st February 1999 (according the the newsgroup archive), though the date on the web version says "revised April 1999". While there have been a couple of version updates of the game stars since those dates, though the best of my knowledge none of these updates changed the internal formula used within the game.

As such the article is still valid for the current version of stars.

## <u>Question 1b (ii) – Literature Search</u>

As a long time member of the stars community I have previously read the majority of all strategy and technical articles on the subject of stars. Additionally I have read the rec.games.computer.stars consistently for a period of over 2 years discussing the various issues associated with the game, this is combined with personal research and testing in the game. This approach requires a lot of reading over a long period of time, but has given me expert knowledge and understanding of the game and its inner workings.

The second method used is for subjects that I am not so familiar with, such as Java reference. It involves doing various web-searches and Usenet archive searches to find website and articles on the subject I am looking for. The process is fairly hit and miss and often requires quickly scanning through many different web pages and refining search terms before finding exactly what I need. When I find a website or article that is relevant to my question, then I usually bookmark it for later reference.

The first approach requires becoming familiar with all the major resources for a particular subject area. Knowing which sites contain useful and well organized information and presented in the formats that are needed (ie tutorials or reference guides or quick tips). It can take a while to become familiar with the various resources in a subject area. In this method, I select a website that has the data in the format that I need it and then navigate though the sites own structure.

The second method is used when I am unfamiliar with the useful resources in a subject area. It involves doing repeated google searches, both on web pages and the Usenet archives, repeatedly refining the search terms and quickly scanning through many links trying to locate a relevant page. This method is far more hit and miss than the first, often resulting in having to go through several irrelevant pages before finding what I need. It does have the side advantage of exposing me to “random” information that I may be able to make use of at a later point in time.

While the first method usually produces better results quicker, especially for reference style information, but it requires a pre-existing knowledge of quality online resources, which are often only found though the second method. The second method however is often better for troubleshooting obscure problems or issues that you are trying to resolve.

## Question 1c – What Skills do you need?

I will need several skills to see this project through to completion.

I have almost reached the completion of the UML design phase. Once this is reached I can automatically create a skeleton set of Java classes to match the UML specification.

I will then need to code each of the function in Java. In accordance with the principals of XP (Extreme Programming) it will require designing and writing a test suite for each function, followed by coding the function until it passes the test. During the coding phase, it may require that I add functions functions or otherwise refactor the design a little.

The creation of a user interface will also be needed for the project. While I already have the majority of the user interface designed from the initial MARS project it will also need to be reimplemented in Java.

Assuming that the function specification and test suite are suitably well written, the need for further testing of the program should be minimized. At this stage the program should be ready for an alpha release to the stars community. The major item to be tested will be on the issue of if the program accurately replicates the results of a battle within the game stars, which is a test of the accuracy of the original specification.

In order to release the program, the source code would need to be made available on a publicly available CVS (Concurrent Versions System) tree (such as sourceforge.net). Additionally I will be required to package the software into an easily installable format. along with a compiled version with instructions for use.

Once the initial release had been achieved, future program development would assume a more iterative and modular approach. New features would be considered, then designed and coded. As well as reported bugs being fixed.

Condensing the above into a list of skills, I will require the following:
- Java coding
- XP (Extreme Programming) Test suite development
- CVS (Concurrent Versions System) usage
- Packaging of a software product (creating an installer etc)

With Java coding, the main issue will be re-familiarizing myself with the java class libraries, the biggest two that I will need to revise are the collection classes as well as the swing/awt classes. To do this, I will rely on the java class documentation as well as referencing online tutorials and guides.

XP (Extreme Programming) test suite development will mainly be practice of writing unit cases and working with a test suite on my own code that will give me the skills needed for this area. I will also be able to do some additional reading on the subject that will assist me as well as looking through examples of test suites within other open source programs.

With the CVS usage and software package creation, these are two skills that while not overly complex, will require further research and practice to properly master.

Looking back, the design phase has cleared up most of the "confusion"associated with the coding aspect of the project. Looking ahead, I do not see any obstacle that will be impossible to overcome. The biggest issue I actually foresee with the above issues is that in the areas I am unfamiliar with, I will have to spend time researching and then becoming familiar with them, though this will have a high educational value and speed me up in future projects. Due to uncertainties about the future, I believe that my biggest obstacle will be not be a technical one (as they can be overcome given enough time and effort), but rather a lack of time to have the project fully developed according to the schedule of the course. While it is difficult to predict how much this will affect the course of the project, the obvious solution is to redefine the how "fully developed" the project needs to be by the project deadline and move parts of the functionality into the extended feature set for future development.

## Question 2B (a) - Describe how your project has evolved

As of TMA01, I had developed a rudimentary UML Class Diagram. Since then, I have put a lot more work into developing the UML diagrams. I have a fairly comprehensive set of Activity diagrams detailing most of the functionality I intend to implement for the first release. I have additionally refined and refactored my Class Diagram. I have started on a few collaboration diagrams, though have not progressed far on them. The project is nearing the end of the design phase and nearing the coding stage.

Attached are copies of the UML diagrams that I have created for the project to date.

Prior to this project, I had written and researched “Guts of the Battle Engine” which is the core specification document for my project. This can be found from pages 20 to 23.

## Question 2B (b) - Changes to the Project based on Tutor's Feedback

In response to my tutors comments, the major issue he seemed to focus on was that I had not adequately or extensively enough explained all the background, purposes and logics behind my project. I partially attribute this to myself overly focusing on trying to summarize the project proposal and meeting the word counts.

Additionally I have created an account on sourceforge.net for hosting my project, (http://sourceforge.net/projects/mars4stars/)

I have not made any material changes to my project, as I had planned it, though I will update and expand my description of it.

Attached below (pages 11 to 19) is my revised project proposal, with additions/modifications highlighted in bold. I have expanded and/or reworded each of the sections that my tutor had commented on or marked out in red. I am also attaching a photocopy of my TMA01 assignment that was returned to me from my tutor.

I have the layouts for most of the user interface already designed from my previous MARS project, though these will need to be recoded in Java.

## Question 2B (c) - What work do you intend to carry out next

The project as it stands is nearing the end of the design phase, with the Class Diagram and Activity Diagrams completed (subject to minor refactoring)

To complete the design phase will require the creation of collaboration diagrams to complement some of the activity diagrams. I will not need to create collaboration diagrams for each activity diagram, but simply reserve them for the cases where there are interactions between several classes and multiple instances. I do not intend to spend additional much time trying to "perfect" my UML diagrams or making them conform to the standards. I will consider them good enough when they have helped me clear up enough of the "confusion" and "unknowns" within the design to allow me to start coding at a lower level without having to constantly think back to the higher level.

Hopefully this should not take very long as I am fairly eager finish the design stage and to start doing some "real" programming. Additionally I am starting to get worried that I might almost be spending too much time on the design phase.

The next major stage will be implementing the designed system. While I will have the help of a code generator within Umbrello UML to generate skeleton code matching the specification, I will need to implement Java code for each of the functions. I will try to follow the "Extreme Programming" (XP) programming methodology, especially in relation to unit tests and test suits. This involves writing a unit test for the part of the function I intend to implement before actually implementing the function.

While writing the code, I may also notice the need for refactorings withing the design which I would then alter to suit.

I will need to additionally design and implement a User Interface in Java for the user to control the program. I already have screen layouts designed from my previous MARS project, though I will need to recode them in Java. These User Interfaces are largely based on the user interface within stars with a few extensions to match the additional functionality within my program.

Once the existing design is fully coded and I have a program that compiles and is validated by the test suites, I will almost ready for my first public beta release.

After this the program can adopt a far shorter design-code-release cycle and development will mainly focus on bug fixes and new features as most of the core functionality should be complete. It will also be at a stage where I would hope that other members of the stars community would contribute to the code.

I have kept a project timesheet (see below), though there have been several instances where I have failed to note my exact hours or even failed to note that I worked on the project on particular days. As a result I would probably estimate that I have only recorded half to two-thirds of the hours I have spent directly working on the project. Additionally I have not documented time spent researching or in skills refinement. So given the 55 hours recorded, I would estimate that I have actually spent between 80 and 110 hours on the project since March. This is less than half the hours that I had budgeted for this time period.

According to my original Gannt chart, I would have finished the design work by the end of May and be starting on the implementation stage (Though I did have TMA02 down for being done during June). Also in further study of the development of test suites I have learned that it should not be developed as a separate stage, but rather at the same time the code is being written.

While I am slightly behind my original schedule for the start of June, it is not a significant setback. Given the overestimation of time it would take to get to this point I have reevaluated by Gantt chart and reduced the time allowance for most of the items. I have reduced the total time estimation from 750 hours to 400 hours.

*Revised Gantt chart of project schedule*

| **Description** | **Budget** | **Mar** | **April** | **May** | **June** | **July** | **Aug** | **Oct** | **Sept** | **Nov** | **Dec** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Project Management | 30 | 10 | 5 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |
| Skills Refinement | 50 | 10 | 10 | 5 | 5 | 5 | 5 | 5 | 5 | | |
| Project Specification | 18 | 13 | 5 | | | | | | | | |
| Design Class Model | 10 | 5 | 5 | | | | | | | | |
| Design Flow Diagrams | 45 | | 10 | 25 | 10 | | | | | | |
| Design Use Case Models | 2 | | 1 | 1 | | | | | | | |
| Design User Interface | 10 | | | | | | 10 | | | | |
| Design Test Framework | 0 | | | | | | | | | | |
| Implement Test Model | 25 | | | | 10 | 10 | 5 | | | | |
| Implement Objects | 40 | | | | 10 | 15 | 10 | 5 | | | |
| Implement User Interface | 15 | | | | | | | 10 | 5 | | |
| Debug Classes | 20 | | | | 5 | 5 | 5 | 5 | | | |
| Debug User Interface | 15 | | | | | | | 10 | 5 | | |
| Test Against Specifications | 40 | | | | 2 | 2 | 2 | 2 | 2 | 20 | 10 |
| Document End User System | 10 | | | | | | | | | 10 | |
| TMA01 Submission | 10 | | 10 | | | | | | | | |
| TMA02 Submission | 10 | | | 10 | | | | | | | |
| TMA03 Submission | 10 | | | | | | 10 | | | | |
| Project Submission | 30 | | | | | | | | 30 | | |
| TMA04 Submission | 10 | | | | | | | | | | 10 |
| **TOTALS** | **400** | **38** | **46** | **43** | **44** | **39** | **49** | **39** | **49** | **32** | **21** |

## MARS4STARS project timesheet 2004

| Date | | | | Time |
|---|---|---|---|---|
| 20th March | Class Diagram | Object Model | Paper | *4h 0min* |
| | Use Case | Fire Weapon | Paper | |
| | Use Case | Calculate Ship Cost | Paper | |
| 7th March | OU Project | TMA 01 | | *2h 0min* |
| 9th April | Manual | Umbrello UML | | *0h 20min* |
| | Class Diagram | | Umbrello UML | *2h 0min* |
| 10th April | Class Diagram | | Umbrello UML | *2h 0min* |
| 11th April | Class Diagram | | Umbrello UML | *2h 0min* |
| 12th April | Class Diagram | | Umbrello UML | *2h 0min* |
| 12th April | Activity Diagram | Main Battle Sequence | Umbrello UML | *2h 0min* |
| | Design | Listed Special Rule List | Paper | |
| 13th April | Activity Diagram | Main Battle Sequence | Umbrello UML | *0h 57min* |
| 14th April | Activity Diagram | | Umbrello UML | *3h 0min* |
| 15th April | Activity Diagram | | Umbrello UML | *8h 0min* |
| 16th April | Activity Diagram | | Umbrello UML | *4h 0min* |
| 14th May | Activity Diagram | | Umbrello UML | *0h 30min* |
| 15th May | Activity Diagram | | Umbrello UML | *5h 30min* |
| 17th May | Activity Diagram | | Umbrello UML | *0h 15min* |
| 21st May | OU Project | TMA 02 | Paper | *7h 0min* |
| 22st May | Class Diagram | Object Model | Umbrello UML | *5h 0min* |
| 21st May | OU Project | TMA 02 | Paper | *5h 0min* |
| | | | Total Time | 55h 32min |

# Revised Project Proposal from TMA 01

**Project Title:**
MARS4STARS (Multiple Assault Resolution Simulator for Stars)

**Project Aims:**

This project has two primary aims;

The first is a practical aim, to design and code a computer program that can accurately simulate the outcome of battles in **computer game stars (to the bug level)** and have extended functionality to allow ship designs, ship counts and battle orders to be easily tweaked and the battle replayed. It will also require a user-interface that is familiar to players of the game. **The purpose of this is facilitate stars players to better test ship and fleet counter-designs within the game, additionally it will enable players to predict the outcome of potential battles and thus adjust their plans accordingly.**

The second is an educational aim, which is to increase my knowledge and experience of computer programming in Java, UML modeling, object oriented design and in using the available development tools. I believe that my project proposal is sufficiently large and complex enough to enable me to become competent in using these skills and to be personally willing to call myself fluent in them.

**Project Background:**

Stars is **a turn-based, conquer the universe, computer strategy game of the 4X genre (eXplore, eXpand, eXploit, and eXterminate)**. You start with a single planet with small population, each turn your population grows, you can build factories/mines, send colonists to other planets, research new technologies and design and build warships. The game is usually played by email, with each player giving orders to each of his planets and fleets, then sends his turnfile to the game host. When the game host has all the turn files in, or a set time limit is up (eg after 24 hours), the host generates the game, and sends each player the turnfiles for the next turn, which tells the player what happened (ie where is ships are, how much his population has grown and the results of any battles that have taken place).

Battles between ~~enemy~~ fleets are calculated by the computer during turn generation. Battle**s** takes place **on** a 2 dimensional 10x10 grid. **There is no user intervention during the battle, and the movement and target priorities are controlled through set mathematical formulas and a fairly simple Artificial Intelligence (AI).** The player's only control over the battle is to specify which ships he sends into battle and by specifying battle orders, **which affects how the AI controls the ships movement and which ship types will be targeted first and which races are to be considered as friendly or hostile.** After the battle, the player can see the outcome of the battle **through the "Battle VCR". This is an in-game window that shows the pre-recorded outcome of the battle,** including how ships moved, shot and took damage.

**Word Count 278**

## Current solutions to the targeted problem

**While there are several helper applications in existence for calculating the formulas used in other areas of stars such as how much fuel as ship will need to go from point A to point B or how many bombs you need to drop on a planet to kill all its population. There is currently no program available that can simulate the outcome of a battle.**

**There are some spreadsheets that will calculate simple battles, with only one design on each side, while these are useful for creating rules of thumb regarding the ratio of ships needed for victory, they are insufficient for simulating more complex battles and make simplifying assumptions regarding the how the AI will move the ships and other factors.**

**For more complex battles, the usual solution has been to create a testbed game of stars, where you control all the players in the game and build ships of the designs you wish to test. There is a generic "battlesim" testbed game available for download, that has a large selection of race combinations, with each player at maximum tech and having access to the special "Mystery Trader" components (that can only be acquired through special events).**

**The "battlesim" testbed saves a lot of time in creating a custom testbed to exactly match the conditions needed for testing. Though a player must still build ships and move them them into position for two or more races (which must be done in two separate instances of the program). This process then needs to be repeated for each re-simulation that changes a few variables. Each time ships have to be built or moved, a new turn must be generated (though a special host program) and all the turns reloaded.**

**Additionally the as the races in the "battlesim"testbed have the maximum levels of tech it does not account for miniaturization (components get cheaper as a players tech levels increase. The targeting algorithm seeks the highest cost ship with the least defenses)**

**My program will make testing easier and less time consuming. A player will be able to specify the exact races to be tested against and all the designs for all players in a single window. Changes that affect the battle, such as the number of ships present or battle orders can also be changed quickly within the UI. The program will enable the same battle to be run multiple times in succession to calculate if random elements (such as the accuracy of a missile) will affect the outcome of a battle. It should also be possible to automate the calculation of some optimal values, such as how many ships would be needed to secure a victory with minimal losses, or which battle orders would generate the most favorable outcome.**

**Word Count 461**

**<u>Project Description:</u>**

**The project is designed to run interactively on a desktop computer. It is intended that it will be cross-platform and operating system independent. It is not an embedded system and will not be directly interfacing with any external hardware.**

To create a battle simulator will require cloning of the battle VCR component of the computer game stars **(to the bug level)**. This component is dependent on many of the data structures and rules within the game, which will also need to be implemented. **While some are explicitly described within the game and its help file, the majority have been found through reverse engineering by the stars community. This project does not need to match the internal structure of stars (which was programmed in C without object-oriented techniques) though it must produce the same results.**

The purpose of this program is to simulate battles and aid in the designing/refining of ship designs, fleet counts and battle orders. As such I will need to add in advanced analysis features and also extend the user interface to display any related data that is would be useful to analyzing the battle (even if it is not shown in the stars GUI).

From an initial analysis, the program will require the following key components::

– An object model consisting of all the various objects and data structures required to simulate the battle engine and store the user's input..

– A component that can initialize the object model from a data file.

– A component that can write the object model to a data file.

– A component that can simulate the rules of the battle engine**, such as calculating if a ship is in range to be shot and how much damage it would take etc.**

– A component that can simulate the computer AI **for deciding how a ship should move and who is should shoot.**

– A GUI for entering data, running battle simulations and viewing the results.

– A command line interface for those wishing to run the program via a script

– Extra components to facilitate advanced analysis features as required

The usefulness of this program will be dependent **on how close it predicts battle outcomes compared to the exact same scenario within the game stars itself. This is really a test of the accuracy of the specification. This will have to be tested though a creating a testbed game within stars and directly comparing the results of the same battle within the two programs.**

I do not expect reverse engineering as being a major task in the project, as most of the formulas and game logic have already been worked out and documented by members of the stars community.

I have previously collated most of this this information, plus some additional research into the logic behind the battle AI, into an article titled “The Guts of the Battle Engine”. This article will become the main specification document for the project.

The main test specification will be a comparison of the battle results with the game stars. The testing phase may show up inaccuracies in the specification document, which would then need to be revised.

**Word Count 519**

**<u>Proposed Future Project Extensions:</u>**

Time permitting, it may also be possible to add in advanced features, such as running the same simulation multiple times to see how far the battle outcome is affected by randomness, also a feature to optimize a ship design, based on cost and battle efficiency, against an existing enemy design.

Once the project is ready for beta-testing it will be publicly released under the GPL (GNU General Public License - a free software / open-source **and copyleft** license **[all modifications must also be made under the same license]**) . This is both for moral and practical reasons. This will not affect the requirement that this project will be all my own work, as I will be the sole developer until a working **beta version is released**, and very likely be the sole developer after it is released. Should I include any code from others, it will clearly marked as so.

As the project will be cloning significant sections of the game stars, it may at some point in the future be expanded to clone the entire game. Additionally such an expansion may wish to implement different or even multiple component data, battle engine rules and computer AIs.

Thus the project's design and coding should be, where possible, both fairly modular and allow for dynamic component substitution.

**Word Count 215**

**Combined Word Count 1473**

**<u>Project Tools and Resources:</u>**
As part of a separate project to fully learn and switch over to the GNU/Linux operating system and open source free software in general, I will be developing my project on a GNU/Linux system. I have considered using the M301 software, JBuilder and Rational Rose.

I the case of JBuilder, while the course CD does have a Linux version, the software is not free or open source, it is an IDE that is specific to the Java language, and my OU software license only covers educational use.

In the case of Rational Rose, there is a Linux version on the market, but it not included with with the course CD. Additionally, my license for the software is time locked and I will not be able to use the software beyond the end of the year.

For my project I will be using KDevelop as my IDE, this is open source and free software, supports multiple programming languages and integrates with various other open source development tools which I additionally wish to become familiar with.

For my UML modeling I will use Umbrello UML v1.2, which is also open source.

While I do not have any previous experience using these tools, I do not believe that becoming familiar with them will pose a major problem, as I already have an understanding of the key concepts behind there use, and there is a wide variety of web resources documenting their use. Additionally these tools do not come with any license restrictions that would prevent me using them in future projects.

Beyond the development tools, I will require a copy of the game stars for testing purposes. I already have a copy of this game and I am very familiar with it.

Through using the output of another program (StarsEd by Stuart Douglas) I have extracted all the component data from the stars binary to a **comma separate value** text file, **with each of the component's values specified in fields**. While this data could have been collated manually, this resource reduces the workload for the project, additionally, by using this as a data source, my program would become usable for modified versions of stars, by using the appropriate data file.

**A sample line from this file is below, which lists the component type (2 = beam weapon) and number of component, followed by its name, 6 fields of technology requirements, items mass, 4 fields of its cost to produce, its icon number followed by its range in squares, damage done to enemy, initiative for firing order and the type of beam weapon it is. The remainder of the fields only apply to ship hulls.**

```
2,2,"X-Ray Laser",2,0,3,0,0,0,0,1,6,0,6,0,29,1,16,9,0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
```

I have also acquired a set of bitmap images that have been extracted from the stars binary, this will simplify the task of  making the program look visually similar to the original.

## Project History:

I originally attempted a similar project in 2001, programming it in C++ without any formal design. I managed to implement the GUI, object model and initialization from a text file. The project was worked on for several months on and off in my spare time and then resumed again briefly a year later but is still incomplete. However I did not keep any records of how much time was actually spent developing the project.

Counting the lines of code in the project reveals a total of 3170 physical lines of code (ignoring comments and blank lines - using SLOCcount). My estimation is that this project was about two thirds complete. While this project was left incomplete, it did make me fluent in C++ programming and confident in using object oriented principles.

The project was left incomplete, mainly due to lack of time, though also the lack of a formal design combined with the size of the project was making it difficult to think with the object interactions and resume coding after having taken a break from coding.

## Project Schedule:

I am using the size of my previous project as a guideline and assuming that coding in java requires a similar number of lines of code to C++. If the previous project was two-thirds complete, then that would suggest this project will comprise of about 4750 lines of code (rounding to the nearest 250 lines of code).

Though I will also give high and low estimates as well. The low estimate would be based on the new project being roughly as large as the existing code for my previous project which would suggest a rounded figure of 3250 lines of code.

The high estimate would be based on the previous project only being half complete, which would 6500 lines of code.

I will use the COCOMO II model to estimate the amount of time and effort I will need to design and implement the MARS4STARS project. I will use the default values for an organic project. Though as I will be the sole developer on the project, the optimal duration figures do not make logical sense. For the above KLOC figures the COCOMO model would estimate:

| | |
|---|---|
| (3.25 KLOC) | Effort = $2.4 \times 3.25^{1.05}$ = 8.27 person months. |
| (4.75 KLOC) | Effort = $2.4 \times 4.75^{1.05}$ = 12.30 person months. |
| (4.50 KLOC) | Effort = $2.4 \times 6.50^{1.05}$ = 17.13 person months. |
| (Average) | Effort = 12.57 person months. |

I can see that my project is an ambitious one, though I believe that I can at least produce a working model of it within the required required timescale of the TM420 course given extra investment of time beyond the suggested 260 hours. However I also believe that the project needs to be of this sort of size in order to fulfill its second objective of making me fluent with program design and java coding.

However there are also several factors to consider that would reduce the effort required to complete this project. Large parts of the design have already been worked out, though not yet documented, especially in regards to the object model, user interface and initialization procedures. Also as I am a sole developer, there is very little in the way of communications overhead. These factors could possibly reduce the COCOMO estimate by a factor of 2, giving my estimate of about 6 person-months to complete, or 750 hours (based on a 30 hour working week). I have scheduled out a time plan using a Gantt chart, though this chart will need to be reassessed during the project as due to inexperience, some of my estimations may not be accurate.

*Gantt Chart showing time estimation for project.*

| Description | Budget | Mar | April | May | June | July | Aug | Oct | Sept | Nov | Dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | |
| Project Management | 30 | 10 | 5 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |
| Skills Refinement | 50 | 10 | 10 | 5 | 5 | 5 | 5 | 5 | 5 | | |
| Project Specification | 20 | 15 | 5 | | | | | | | | |
| Design Class Model | 10 | 5 | 5 | | | | | | | | |
| Design Flow Diagrams | 40 | | 40 | | | | | | | | |
| Design Use Case Models | 25 | | 15 | 10 | | | | | | | |
| Design User Interface | 10 | | | 10 | | | | | | | |
| Design Test Framework | 20 | | | 20 | | | | | | | |
| Implement Test Model | 40 | | | 40 | | | | | | | |
| Implement Objects | 100 | | | | 40 | 40 | 10 | 10 | | | |
| Implement User Interface | 40 | | | | | | | 30 | 10 | | |
| Debug Classes | 100 | | | | 20 | 30 | 40 | 10 | | | |
| Debug User Interface | 40 | | | | | | | 20 | 20 | | |
| Test Against Specifications | 150 | | | | 10 | 10 | 10 | 10 | 20 | 40 | 50 |
| Document End User System | 5 | | | | | | | | | 5 | |
| TMA01 Submission | 10 | | 10 | | | | | | | | |
| TMA02 Submission | 10 | | | | 10 | | | | | | |
| TMA03 Submission | 10 | | | | | | 10 | | | | |
| Project Submission | 30 | | | | | | | | 30 | | |
| TMA04 Submission | 10 | | | | | | | | | | 10 |

The major milestones will be:

1. The completion of the design stage including the UML models / flow charts / and use case models and test suite.

2. A coded version of the design that compiles and is validated by the test suite

3. A system that has been fully tested against the stars program is can accurately predict battle outcomes.

In the eventuality that the project slips significantly behind schedule, then it would be possible to only aim for completion of the second milestone by the end of the year. Additionally some of the non-core functionality could be delayed in implementation until after the OU course has finished.

## Literature Search

**As I am already very familiar with the articles and other literature on the subject of stars, I did not have much to do in the way of "searching" for it, but rather simply locating copies of things I knew I would need to reference. Below is a small selection of documents that I will be referencing, relating to the game stars, during my project.**

*** Guts of the Battle Engine by James McGuigan ***
**http://www.starsfaq.com/battleengine.htm**
**This is an article I wrote and researched myself specifically with this project in mind. It contains a detailed explanation of the internal workings of the battle engine, to the best of my abilities of understanding it. It will form the key specification document for my project.**

*** Targeting Order in Battles by Art Lathrop ***
**http://www.starsfaq.com/articles/sru/art201.htm**
**This contains a mathematical formula for calculating the order in which the stars battle engine targets ships. It has been widely used and tested and I believe it to be fairly accurate. It is one of the key elements that I will need to implement to get an accurate simulation of the stars battle engine.**

*** Official Stars Help File ***
**ftp://crisium.com/stars/patches/e27ihelp.exe**
**Its primary focus is on describing the various options and how to play the game. However it does contain the occasional references to the games internal mechanics and formulas. While these are occasionally incorrect or vague it will still useful as a reference for certain items that I will need to implement.**

**END OF PROJECT PROPOSAL**

# Guts of the Battle Engine

http://www.starsfaq.com/battleengine.htm

**Author:** James McGuigan
**Written:** 23 Jan 2002

Here are the guts of the battle engine as I understand it from both experience, observation and the help file (please pull me up on any points I get wrong)

For a battle to take place 2 or more fleets (or a fleet and a starbase) must be at the same location and at least one of the fleets must be armed and have orders to attack ships of the others race (the type of ships involved doesn't matter). If are race has a fleet present at a location where there is a battle, but doesn't have orders to attack any of the other races there and none of the other races present has orders to attack it then it will not take part in the battle (and can not benefit from potential tech gain).

Each ship present at the battle will form part of a token (AKA a stack), it is possible to have a token comprised of just a single ship. Tokens are always of ships of the same design. Each ship design in each fleet will create a token, splitting a few ships off to form a second fleet before the battle will create a second token on the battle board.

The battle grid is made up of 10 squares by 10 squares. Each token is in a single square, there can be more than one token in the same square. There is an limit of 256 tokens per battle event for all players involved, if this limit is exceeded, then excess tokens will be left out (those created from fleets with the highest fleet numbers), in such a case each player will have an equal number of tokens, each player will be guaranteed to get their "share" of the available token slots (ie in a 4 race battle 256 / 4 = 64 token slots), if a race doesn't use up all their "slots" then they are shared equally between the other players.

Each battle is made up of rounds. There are a maximum of 16 rounds in each battle. Each round has two parts, movement and shooting. Each token has a speed rating, and will be able to move between 0 and 3 squares in a single turn. If a token has a fractional speed rating then they will get a bonus square of movement every set number of turns. a 1/4 bonus means an extra square of movement on the first round and then on every fourth round after that starting with the fifth. A 1/2 speed bonus gets a bonus square of movement every other turn starting with the first, and a 3/4 speed bonus gets a bonus square of movement for the first three rounds of every 4 round cycle. The order of movement is this, each token with 3 movement squares moves a single square, then each token with 2+ movement moves a single square (if it had speed 3 then it would move for its second square) and then all ships with at least one square of movement move again. At each stage the ships with the most weight will move first though there is less than a 20% difference in weight then there is a chance that the lighter ship will go first. The smaller the weight % difference the greater the chance of the lighter ship going first.
Each token has an attractiveness rating. This is used in both working out where ships

move to and which ships are shot at first. The essence of the formula is cost / defense. A ship will have different attractiveness ratings verses different types of weapons (beams, sappers, torpedoes and capital missiles). Cost is calculated by summing the resource and boranium costs of the ship design used (iron and germ costs don't affect the attractiveness rating). Defense is calculated by the shield and armour dp modified by the enemies torpedo accuracy (after base accuracy, comps and jammers are worked out) if defending vs torps or capital missiles, the effects of double damage for unshielded targets vs capital missiles and the effects of deflectors against beam weapons. The attractiveness rating can be change during the course of the battle as shields and armour deplete. Attractiveness doesn't take into account the one missile one kill rule, thus chaff has become a fairly effective tactic.

Battle orders are comprised of 4 parts. A primary and secondary target type, legitimate races to attack and the tactic to use in battle. Ships will only attack tokens belonging to legitimate target races, however if another race present has any ships (including unarmed ships) with battles orders to attack your race then that race will also be considered a legitimate target. When attacking ships will try and shoot the most attractive ship of a type listed as a primary target and if no ships are available which are primary targets then the most attractive ship of a type listed as a secondary target will be targeted. Ships which are not listed as primary or secondary targets will not get shot at, even if they are shooting back.

There are 6 different battle orders which determine the movement AI of the ships in battle, the movement AI is applied each time a ship wants to move a square on the battle board.:

**Disengage -** If there is any enemy ship in firing range then move to any square further away than your current square. If you are in range of an enemy weapon but cannot move further away then try move to a square that is of the same distance away. If you are in range of the enemies weapons and cannot move away or maintain distance then move to a random square. If you are not in range of the enemies weapons then move randomly. Also you will try and disengage which will require 7 squares of movement to be clocked up before you can leave from the battle board.

**Disengage if Challenged -** Behaves like Maximise Damage until token takes damage and then behaves like Disengage.

**Minimise Damage to Self -** (Not 100% sure on this one) If within range of an enemy weapon then move away from the enemy (just like Disengage). If out of range of the enemies weapons or cannot move away from the enemy then try and get in range of the best available target without moving towards the enemy.

**Maximise Net Damage -** Locate most attractive primary target (or secondary if no primary targets are left). If out of range with ANY weapon then move towards target. If in range with all weapons them move as to maximise damage_done/damage_taken. The effect of this is if your weapons are longer range then try to stay at maximum range. If your weapons range is the same then do random movement while staying in range. If your weapons are shorter range and also beam weapons then attempt to close in to zero range.

**Maximise Damage Ratio -** As Maximise Net Damage but only considers the longest range weapon.

**Maximise Damage -** Locate most attractive primary target (or secondary if no primary targets are left). If any of your weapons are out of range of that token then keep moving to squares that are closer to it until in range with all weapons. If using any beam weapons (as they have range dissipation) then attempt to close to 0 range. If just using missiles or torps and in range then move randomly to a squares still in range.

Note that there is a bug when fighting starbases, the battle AI doesn't count the +1 range bonus when calculating movement. This mainly applies when your ships are attempting to get out of range of the enemy, so vs starbase with range 6 missiles, your ships will move to distance 7, the movement AI won't calculate that they are still in range even when they keep getting shot at.

After the movement phase all ships will shoot their weapons, a token will fire all weapons from the same slot in a single shot. The weapon slot with the highest initiative will fire first. If there are two ships with slots of the same init, then the ships will be randomly given a priority over who can fire first (which will stick for the entire battle). The rest of the weapon slots are then fired in init order. Damage is worked out in between each shot and applied to the ships. If ships or tokens are destroyed before their turn to shoot then they won't be able to fire back. The movement AI will go after the most attractive primary target on the board, but if this token is not in range, then the ship will fire on the most attractive primary target within range (or secondary if none available). Starbases have a +1 range bonus to all their weapons (this also gets applied to minefield sweeping rates), though cannot move. The movement AI doesn't take this bonus into account when moving ships to close in on an enemy starbase.

Damage for each shot is calculated by multiplying the number of weapons in the slot by the number of ships in the token by the amount of dp the weapon does. For beam weapons, this damage will dissipate by 10% over the range of the beam (for a range 2 beam - no dissipation at range 0, 5% dissipation at range 1 and 10% dissipation at range 2). Also capacitors and deflectors will modify the damage actually done to the enemy ship. Damage will be applied first to the tokens shield stack and then to armour only when the entire shield stack of the token is down. For missile ships, each missile fired will be tested to see if it will hit, the chance to hit is based on the base accuracy, the computers on the ship and the enemy jammers. Missiles that miss will do 1/8 of their damage to the shields and won't affect armour. For missiles that hit, upto half will be taken by the shields, the rest will go to the armour. For capital missiles any damage done after the shields are taken down will do double damage to the armour. Whole ship kills are worked out by adding up all the damage done to the armour by a single salvo (from a token's slot) and dividing this by the amount of armour each single ship in the token has left (total armour x token damage %). The number of complete ships the shot could kill will be removed from the enemy token, the rest of the damage will divided equally among the rest of the ships in the token and applied as damage. As token armour is stored in 1/512ths (about 0.2%s) of total armour and not as an exact dp figure (shields are stored as an exact figure), there may be some rounding of the damage after each salvo (AFAIK its always rounds up). This fact can be abused by creating lots of small fleet tokens with weak missiles and many slots, where each slot that hits will do 0.2% damage to the enemy token even if each individual missile would do less damage normally (especially the case with a beta torp shooting a large nub stack).

After all the weapons that are in range have fired, the next round begins, starting with ship movement.

The battle is ended when either the 16 round timer runs out, there is only one race left present on the battle board or if there are two or more races which have no hostile intentions towards each other.

After a battle, salvage is created. This is equal to 1/3 of the current mineral costs of all the ships that where destroyed during the battle. This is left at the location of the battle and will decay over time, or if the battle happened over a planet, then the minerals will get deposited there.

Any races that took part in a battle and had at least one ship that managed to survive (either through surviving till the end or retreating beforehand) has a potential to gain tech levels from ships that where destroyed during the battle. For the exact details of the formulas and chances involved see the Guts of Tech Trading <http://www.starsfaq.com/tech_trade.htm>

**END OF GUTS OF THE BATTLE ENGINE**