# Defence Science and Technology Group (DSTG) and Swordfish Computing Project

Proposal: Distributed Decision-Making

**Initial Report**

Group COMPLEX 8

| | |
|---|---|
| a1734056 | Hayden Lee |
| a1734069 | Vinh Nguyen |
| a1743599 | Nathan Van der Hoek |
| a1744852 | Harrison Bagley |
| a1746088 | Daniel O'Connor |
| a1746146 | Patrick Capaldo |
| a1748751 | Sarah Damin |
| a1749935 | Simon Davies |
| a1773841 | Hayley Richardson |

## 1.0 Project Vision

The vision of this project is to create a user-friendly software system which can test different distributed decision-making (DDM) algorithms in a range of specified scenarios with a graphical interface. The scenarios will be varied with a set of parameters that can be changed by the user, in order to test how the DDM algorithms behave under different constraints. The system will have a graphical representation of the running scenario and selected distributed algorithm so that the user can easily visualise the behaviours and outcome of the test. The system will be designed for modularity, allowing the testbed to be easily extended with new algorithms. The system will be tested under a range of DDM algorithms in order to determine the effectiveness of the testbed. A range of criteria will be developed to measure the effectiveness of different algorithms in order to determine which distributed algorithm is most successful.

## 2.0 Customer Q&A

On Tuesday the 2nd of August 2022 at 12pm a kick off meeting was held with the product owner. The purpose of this meeting was to understand the intent of the project. The kick off meeting also provided a forum for which the project group could ask the product owner questions regarding the distillation of project requirements. The questions and answer are tabulated in Table 2.1.

*Table 2.1 showing questions and answers of initial meeting with product owner*

| Question | Answer |
|---|---|
| What is the aim of the Distributed Decision-Making Software Engineering Project? | The aim of the project is to develop a software system and environment that allows for different distributed decision-making frameworks to be tested. Different DDM algorithms should be experimented with, and their effectiveness compared. After its completion, this project is intended for DSTG scientists to use. |
| What specific algorithms would you like us to investigate and compare? | It is up to you to choose the algorithms you wish to apply. However, your solution must be configurable such that many different algorithms could be applied in future. There are several different classes of DDM algorithms such as consensus algorithms. Be conscious of fault tolerance in algorithms; there's no point in comparing one algorithm with fault tolerance and one without if there are no faults in the scenario. |
| What are the specifics of the representative scenario? | Cowboys are in the field and aliens are trying to destroy a target whilst the cowboys are trying to defend it, the cowboys communicate one-by-one to replicate a distributed decision-making system. |
| How do the agents differ from one another? | Initially there are cowboys with pistols and aliens with lasers. Some cowboys have horses and some aliens have spaceships, thus there are four distinct classes of agents. The horses and spaceships influence the agents' speeds and health points. Extensibility could include new classes of agents such as better spaceships. |
| Are we modularising the algorithms and environment around expected inputs/outputs? | Yes, modularity is a non-functional requirement. We need a framework underneath where we can easily add new algorithms thus it must be extensible. |

| | |
|---|---|
| How do you want the inputs to the GUI to be represented? For example, number of cowboys, aliens, etc. | I do not mind how the data is input. It does not matter if the data is input via sliders, text boxes, or otherwise. The only thing that is critical is that the input is configurable. That is, the input may be changed. |
| What tools are we using to implement scrum methodology? | From a communication perspective, you are free to communicate how you wish as long as I am kept in the loop. Slack is the preference as you can implement a slack bot which allows daily scrums to be performed with ease.<br>For version control and code collaboration, GitHub Projects is the preferred software. |
| We understand the cowboys have a fixed range as they can only communicate verbally, what about the aliens? Do they have a fixed communication range? | There is no requirement that specifies alien communication range. Perhaps they have infinite communication range. This ties in with configurability and extensibility of your solution. |
| Is the environment 2D or 3D? Does it make a difference to meeting the objectives if the environment is made in 2D? | Possibly not but you need to investigate whether it does or not. 2D vs 3D may be a key consideration for fault tolerance. If you find it doesn't contribute to physical differences, then it probably isn't necessary. |

The initial meeting with the product owner provided many answers to questions regarding the specifics of the project. Namely, the goals of the project and the representative scenario. The initial meeting also outlined how much of the project is open to our own interpretation. For example, which algorithms we are to implement and compare, and how we alter the agents in the environment. The project group decided upon several further questions to be asked to further refine the project requirements. These questions were asked on Thursday the 11th of August 2022 in a follow up discussion with the product owner and are tabulated in Table 2.2.

*Table 2.2 showing follow up questions and answers with product owner*

| **Question** | **Answer** |
|---|---|
| Is there a specific way you want to measure the outcome and an algorithm's effectiveness? | I'd like to see metrics directly from the algorithms, e.g., how many messages are sent, how many messages failed, etc. I would also like to see metrics from the simulation, e.g., which team won, how many times (if multiple runs), how many agents were neutralised per team, etc. I think those should give me a pretty good idea about the effectiveness for now. |
| What error metrics do you want to be investigated? How do you want errors to be represented? | I'm particularly interested in errors that happen when agents are removed from the distributed network. I'd also like to know if an algorithm terminates and any of the agents aren't consistent with the rest of the network. |
| Will the project group be provided with example input parameters? | The one input parameter you can be sure of is which algorithm I'd like to simulate with. The other input parameters will be determined by the algorithms you choose, and how they're expressed in the simulation. For example, if you need 10+ agents for a particular algorithm, that should probably be an input parameter. |

| | |
|---|---|
| Is this intended be performed on one computer or on multiple as a distributed system? | One computer please! I don't want to make your lives or my life hell. How you implement the distributed nature is up to you, it could be separate processes communicating with each other and sending their individual properties to a simulation process! Or it could be concurrent with threads instead of processes. Or it could be completely lock-step and really, really simple. |

Upon reflection of the kick-off meeting and subsequent follow-up questions, two important lessons have been identified which should improve the effectiveness of future meetings: better question preparation and more detailed questions. Allowing more time to develop questions improves the confidence of covering all the current and important topics, whilst increasing the detail of each question ensures more information is identified from the Product Owner within the same amount of meeting time.

# 3.0 Users

The end user of the software package will be the DSTG scientists. Ultimately, the scientists desire a system that allows them to compare the effectiveness of various distributed making algorithms. When using the software, the workflow of a scientist, outlined in Figure 3.1, would consist of selection of the simulation parameters, selection of the DDM algorithm of interest then running the simulation. Following this, the scientists would then expect to obtain the results of the simulation.

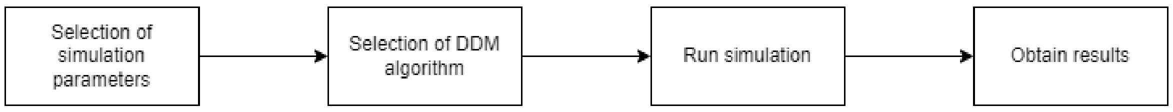

Selection of simulation parameters
Selection of DDM algorithm
Run simulation
Obtain results

*Figure 3.1: DSTG scientist workflow when using software*

In order to effectively satisfy this workflow, the scientists require a system that allows varying the parameters of the simulation to permit testing of different scenarios. The scientists also require a system that can interface to a multitude of algorithms that function in different ways. The system also needs to permit simple interchanging of the DDM algorithms being used. The scientists also need a system that can then run a simulation using the DDM algorithms in the environment. Moreover, the scientists require a system that produces metrics to permit effective comparison of the DDM algorithms. They also require visual representation of the algorithms' performances. Finally, the system needs to be simple and easily used.

## 4.0 Software Architecture

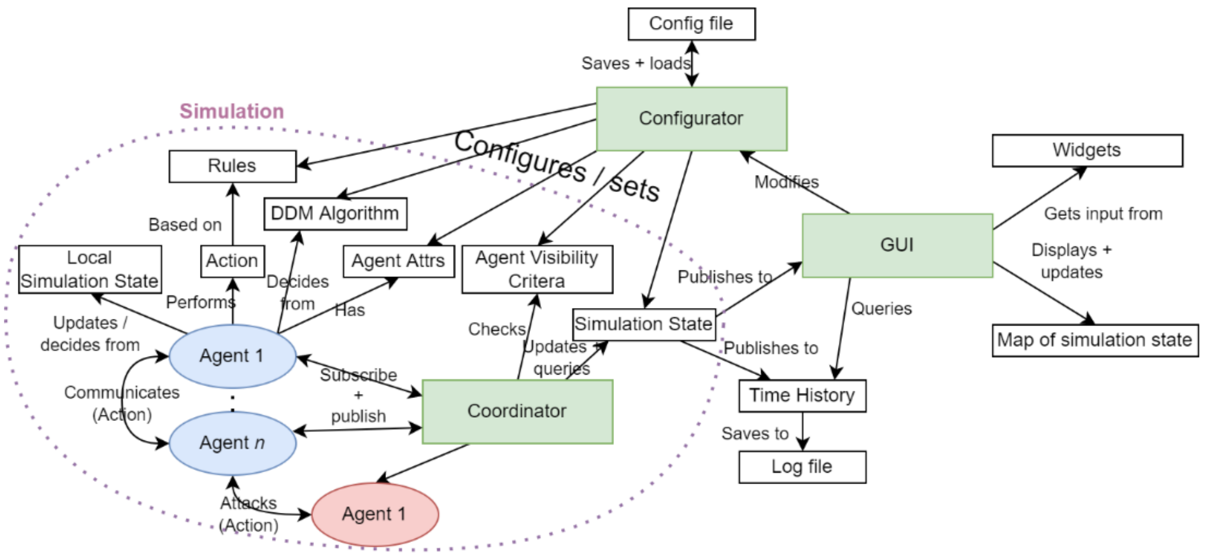


*Figure 4.1: Proposed software architecture*

The proposed software architecture demonstrates different high-level components interacting in different ways and is subject to change with client feedback. To increase the modularity of the code architecture, design patterns are to be utilised which are "typical solutions to common problems in software design" (Refactoring Guru, 2022). The backend runs the simulation (circled in purple dots in Figure 4.1) and is the most significant part of the software. The simulation's state and properties are initialised by the 'Configurator' and simulation updates are facilitated by the 'Coordinator'. Using the Observer design pattern, external components such as the GUI and time history logger are updated about changes to the simulation state. The GUI calls methods on the configurator to update the simulation configuration. This code architecture allows the user to set the DDM algorithms inside the simulation and maintains the separation of the simulation from external components.

The Configurator configures the initial aspects of the simulation; it can be manipulated by the GUI, and it saves and loads configurations to persistent files. The Coordinator also implements the mediator design pattern, informing external components of changes caused by Agents' actions. The GUI subscribes to the Coordinator so it can update a map display whenever a change occurs. The 'time history' also subscribes to the Coordinator and writes every change to file, for a full history of the simulation. The GUI can then query the time history to display the state of simulation at any point in time throughout.

The simulation itself consists of several different interacting components. There are Agents, which have various attributes and DDM algorithm both of which can be configured. The Strategy design pattern will be used to compose Agents of attributes and Actions, which allows for greater code extensibility and polymorphism. The Actions' behaviour and effects are determined by Rules, which can be reconfigured. Each Agent duplicates the initial simulation state and maintains its own understanding of the simulation state, which aids the DDM algorithm in choosing an Action. Agents maintain a list of other Agents that they are aware of and thus can perform Actions on. To facilitate an initial and basic DDM algorithm, Agents will initially be able to take three Actions – *Move*, *Communicate* and *Attack*; they can communicate with friendly Agents and attack enemy Agents. The Observer design pattern will be used such that the Coordinator will subscribe to every Agent and know of every Action that is taken. Based on these Actions, the Coordinator updates a ground-truth simulation state that contains the current state of every Agent and features of the environment. If an Agent performs the *Move* Action, it may result in the Coordinator updating Agents' awareness of each other based on configurable visibility criteria. The Strategy, Mediator, and Observer design patterns will increase code maintainability and extensibility by identifying code interfaces and ensuring it is loosely coupled. It is likely that more design pattern types will be integrated into the code architecture in the future.

## 5.0 Tech Stack and Standards

The tech stack for this project involves three key components: the user, the front-end, and the back-end. The user is the client that will be using the product to conduct experiments regarding the decision-making behaviour of distributed algorithms in a cowboys versus aliens scenario. The client specifies the parameters (e.g., placement of cowboys and aliens (agents), type of algorithm, shooting range of cowboys) of the experiment they wish to conduct by interacting with the front-end. Therefore, to ensure ease-of-use of the experimentation testbed, the front-end must be user friendly and straightforward to utilise to vary parameters. There are two preferable types of user-friendly front-ends that can be developed for this project: configuration file or Graphical User Interface (GUI). A configuration file is a simple text file that lists all the desired parameter values for the simulation in a prescribed order that is known by the parser in the back-end. A GUI uses visual indicators to represent information and actions available to the user and would likely improve the ease-of-use of the front-end. Examples of frameworks that provide efficient GUI development include Java Swing, PyQt, Tkinter, and PyGame, the first of which uses Java as its programming language and the latter all using Python.

The front-end will communicate the choice of parameters to the back-end which then starts running the distributed decision-making simulation until it is completed, or the user ends the simulation. The back-end will be computationally intensive as it will be required to simulate each agent in the environment, their communications between each other according to the distributed algorithm, the destruction and creation of agents, data collection, and data refinement for review by the client. These features are highly likely to exhibit in early versions of the program, yet further additions such as visualising the back-end through dynamically rendered images of the simulation space are also likely required for later versions – further increasing the computational requirements of the back-end. Due to such requirements, a back-end that can efficiently and dynamically render images is desired. Fortunately, agent-based simulation programs, such as AnyLogic, NetLogo, FlexSim, Simio, and Simul8, provide much of the modelling, dynamic visualisation, agent communication, and data logging capabilities in their own self-contained software packages. AnyLogic is a desirable choice over the others as while it is not completely open-source, it does exhibit more widespread community support, is up-to-date, has a free version with little restrictions, can support 2D and 3D visualisation, and supports Java as a programming language meaning that it will integrate nicely with a potential Java Swing-based front-end. The other options exhibit disadvantages such as proprietary native programming languages, limited visualisation, more restrictions within their free versions, and are out-of-date. Furthermore, taking advantage of such software packages allows the developers to avoid rewriting code that already exists and fast-track development to create a better product for the client.

In summary, the tech stack of this program will likely involve the client interacting with a Java Swing front-end GUI (or a configuration file for early versions) to select experiment-specific parameters, which will then queue an AnyLogic-powered back-end to run the simulation and record relevant data until completion (Figure 5.1).

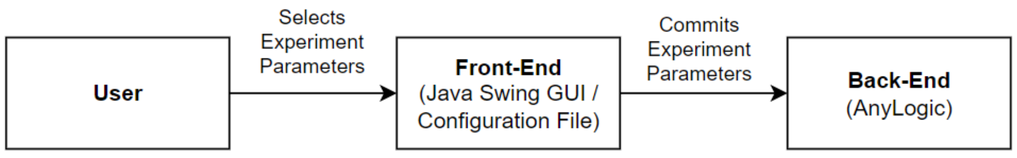

User
Selects Experiment Parameters
Front-End (Java Swing GUI / Configuration File)
Commits Experiment Parameters
Back-End (AnyLogic)

*Figure 5.1: Tech stack for the distributed decision-making software platform.*

### 5.1 Communication

Regarding communication between the development team and the client, a combination of Slack, Microsoft Teams, and GitHub Projects is being utilised. Slack is used for general discussions and file sharing between the development team and the product owner, with capabilities for splitting into smaller groups to allow for more focused lines of effort. Slack also provides a formal medium within which daily stand-ups can be completed in a quicker, automated form via text messages (that in use is the "Dixi App" Slack Bot). This is preferable since the students in the development team are not all available at the same time every working day of the week. Microsoft Teams provides a common video-conferencing medium for weekly development team meetings and fortnightly sprint reviews. GitHub Projects utilises a virtual Kanban board to allow the development team to seamlessly review and communicate progress in completing the product backlog.

### 5.2 Coding Standards

Agreeing upon a prescribed set of coding standards is critical to ensure that code is clean, readable, exhibits minimal errors, is easily maintainable by those other than the original authors, and is efficiently integrated among many parallel developers. It is highly likely that Java will be used as the primary coding language and a comprehensive, highly reliable, and commonly referenced standard is that presented publicly by Google named the "Google Java Style Guide" (Google, 2022). The guide provides highly specific details regarding the style of all aspects of coding in Java such as source file structure, formatting, and naming conventions. Furthermore, the use of Java allows for static code analysers such as PMD to be used to partially automate this process of standardisation. However, developers are required to be conscientious that the suggestions of such analysers do not disagree with the Google Java Style Guide, and if so, must ensure they are changed to agree with the Google Java Style Guide. Additionally, custom rules can be added to the analyser to make it more compatible with the accepted standard.

## 6.0 Group Meetings and Team Member Roles

### 6.1 Group Meetings

Meetings between the Development Team have been scheduled weekly on Mondays from 11am-11:30am in a hybrid form to discuss progress during the sprint, help resolve any blockers, and discuss decisions involving the entire team. The meeting at this time in Weeks 5, 7, 9, and 11 will include the Product Owner and will involve a Sprint Review and Sprint Planning. Additionally, members of the team can informally meet throughout the week to work on the project together.

### 6.2 Sprint Retrospective Meetings

Sprint Retrospective meetings will happen on the Friday before the Monday Sprint Review and Planning meeting, to allow for the team to reflect on the successes and potential improvements of the previous sprint. A specific time has not been confirmed yet as the team members often have conflicting schedules and so times may need to change throughout the semester.

### 6.3 Feedback Channels

The main communication channel between the Development Team and Product Owner is through the slack channel. This channel is used to ask clarifying questions to the product owner while also keeping them involved in the Development Team's progress and working. This facilitates the "fail fast, fail often" approach, by allowing the development team to reach out to the Product Owner quickly and easily to overcome issues. Furthermore, emails and direct slack messaging can also be used for more specific feedback and questions.

**Scrum Masters and Group Roles**

The Scrum Masters for each sprint are described in Table 6.1.

*Table 6.1: Scrum Masters for each Sprint*

| Sprint Number | Weeks | Scrum Master |
|---|---|---|
| 1 | 2 - 4 | Hayden Lee |
| 2 | 5 - 6 | Nathan Van der Hoek |
| 3 | 7 – 8 (incl. mid-semester break) | Vinh Nguyen |
| 4 | 9 - 10 | Patrick Capaldo |
| 5 | 11 - 13 | |

While additional group roles were discussed, the group decided that no formal roles would be held by the rest of the Development Team, and rather tasks would be allocated based on preference and team member strengths.

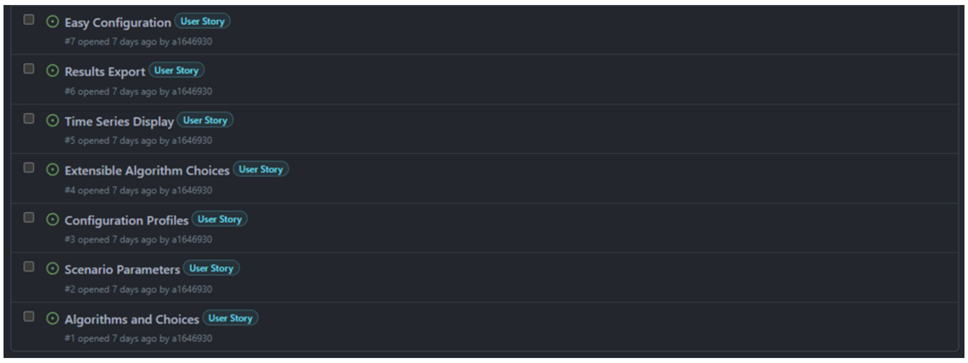
## 7.0 Conclusion

The initial planning of this project, including consultation with the product owner, has resulted in a preliminary understanding of the user needs of the software and a corresponding code architecture. The code architecture is focused on modularity to account for future client vision changes and to synergise with the Agile development process. Currently, the project vision is to create a user-friendly software system for DSTG scientists to test and evaluate the performance of different DDM algorithms. To aid in the Agile development process, a standard Tech Stack and coding practice for the project is defined to ensure project consistency and uniformity. Additionally, to align with the Agile development process, regular structured meetings and easily accessible communications avenues are maintained to ensure upcoming sprint success.

With this initial report, the project team aims to provide insight into current development strategies and provide potential implementation proposals to prompt and aid in further discussions with the client and build mutual understanding. With further client feedback and discussion, the project vision is subject to change and update to align with the client's needs.

## 8.0 Snapshot

### 8.1 Product Backlog and Task Board:


Easy Configuration User Story
#7 opened 7 days ago by a1646930

Results Export User Story
#6 opened 7 days ago by a1646930

Time Series Display User Story
#5 opened 7 days ago by a1646930

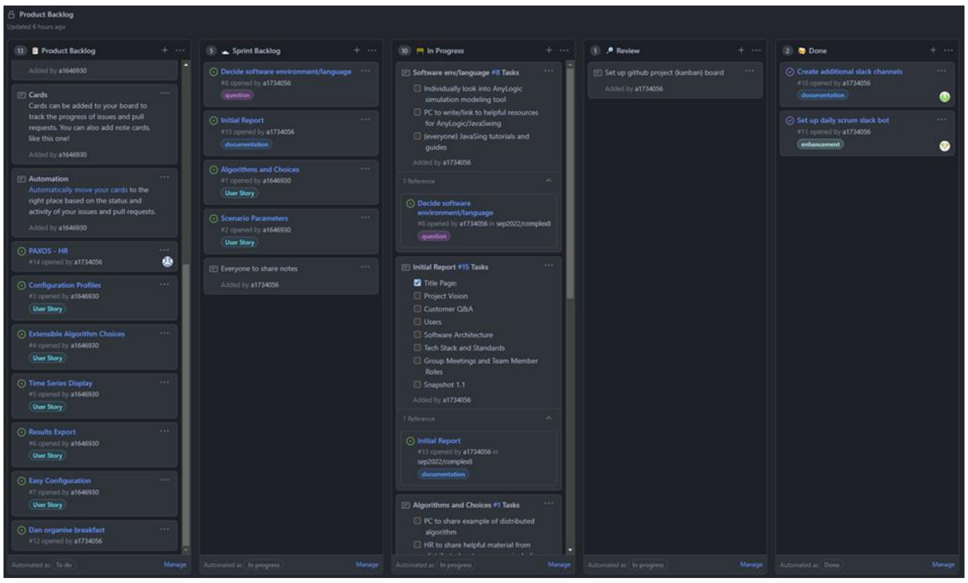
Extensible Algorithm Choices User Story
#4 opened 7 days ago by a1646930

Configuration Profiles User Story
#3 opened 7 days ago by a1646930

Scenario Parameters User Story
#2 opened 7 days ago by a1646930

Algorithms and Choices User Story
#1 opened 7 days ago by a1646930

*Figure 8.1: Product Backlog Screenshot*


Product Backlog
Updated 6 hours ago

13 Product Backlog

Added by a1646930

Cards
Cards can be added to your board to track the progress of issues and pull requests. You can also add note cards, like this one!
Added by a1646930

Automation
Automatically move your cards to the right place based on the status and activity of your issues and pull requests.
Added by a1646930

PAXOS - HR
#14 opened by a1734056

Configuration Profiles
#3 opened by a1646930
User Story

Extensible Algorithm Choices
#4 opened by a1646930
User Story

Time Series Display
#5 opened by a1646930
User Story

Results Export
#6 opened by a1646930
User Story

Easy Configuration
#7 opened by a1646930
User Story

Dan organise breakfast
#12 opened by a1734056

Automated as To do — Manage

5 Sprint Backlog

Decide software environment/language
#8 opened by a1734056
question

Initial Report
#15 opened by a1734056
documentation

Algorithms and Choices
#1 opened by a1646930
User Story

Scenario Parameters
#2 opened by a1646930
User Story

Everyone to share notes
Added by a1734056

Automated as In progress — Manage

10 In Progress

Software env/language #8 Tasks
- [ ] Individually look into AnyLogic simulation modeling tool
- [ ] PC to write/link to helpful resources for AnyLogic/JavaSwing
- [ ] (everyone) JavaSing tutorials and guides

Added by a1734056

1 Reference

Decide software environment/language
#8 opened by a1734056 in sep2022/complex8
question

Initial Report #15 Tasks
- [x] Title Page:
- [ ] Project Vision
- [ ] Customer Q&A
- [ ] Users
- [ ] Software Architecture
- [ ] Tech Stack and Standards
- [ ] Group Meetings and Team Member Roles
- [ ] Snapshot 1.1

Added by a1734056

1 Reference

Initial Report
#15 opened by a1734056 in sep2022/complex8
documentation

Algorithms and Choices #1 Tasks
- [ ] PC to share example of distributed algorithm
- [ ] HR to share helpful material from

Automated as In progress — Manage

1 Review

Set up github project (kanban) board
Added by a1734056

Automated as In progress — Manage

2 Done

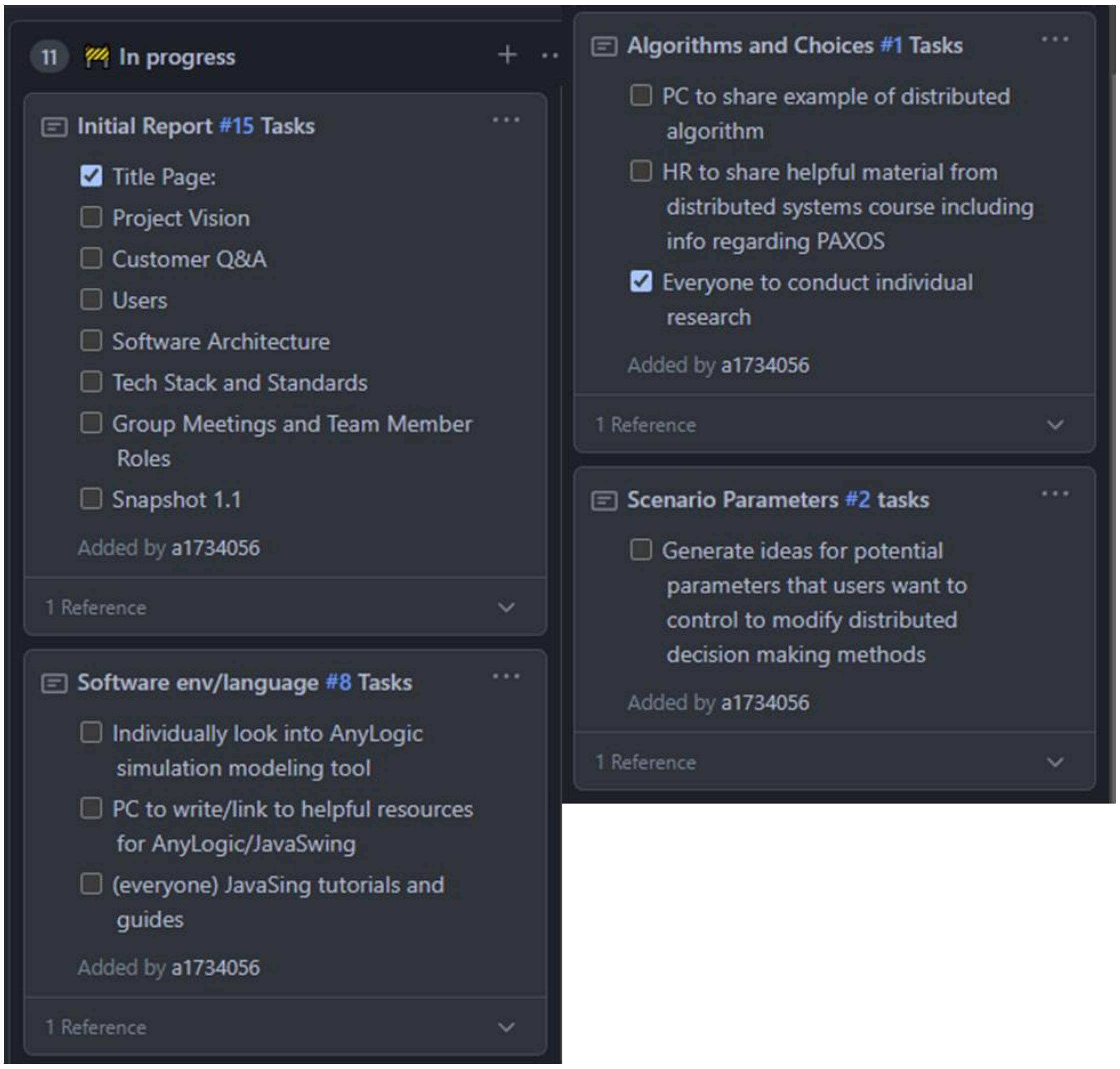
Create additional slack channels
#10 opened by a1734056
documentation

Set up daily scrum slack bot
#11 opened by a1734056
enhancement

Automated as Done — Manage

*Figure 8.2: Task Board Screenshot*

## 8.2 Sprint Backlog and User Stories:



11 🚧 In progress

**Initial Report #15 Tasks**

- [x] Title Page:
- [ ] Project Vision
- [ ] Customer Q&A
- [ ] Users
- [ ] Software Architecture
- [ ] Tech Stack and Standards
- [ ] Group Meetings and Team Member Roles
- [ ] Snapshot 1.1

Added by a1734056

1 Reference

**Software env/language #8 Tasks**

- [ ] Individually look into AnyLogic simulation modeling tool
- [ ] PC to write/link to helpful resources for AnyLogic/JavaSwing
- [ ] (everyone) JavaSing tutorials and guides

Added by a1734056

1 Reference

**Algorithms and Choices #1 Tasks**

- [ ] PC to share example of distributed algorithm
- [ ] HR to share helpful material from distributed systems course including info regarding PAXOS
- [x] Everyone to conduct individual research

Added by a1734056

1 Reference

**Scenario Parameters #2 tasks**

- [ ] Generate ideas for potential parameters that users want to control to modify distributed decision making methods

Added by a1734056

1 Reference

*Figure 8.3: Sprint 1 Backlog*

The selected user stories for the current sprint are Algorithms and Choices (Issue #1) and Scenario Parameters (Issue #2). The Algorithms and Choices user story is described by *"As a typical user, I want to be able to choose at least one algorithm to test, so that I can see how effectively it performs"*. The Scenario Parameters user story is described by *"As a user interested in experimental scenarios, I want to control the parameters of the scenario, so that I can better contrast the results of different algorithms"*.

## 8.3 Definition of Done:

The definition of done is that the following criteria are satisfied:

- Code written and commented
- Documentation written and updated
- Code peer-reviewed
- Documentation peer-reviewed
- Tests written and passing
- Non-functional requirements met (UX, performance, availability)
- Acceptance criteria fulfilled

### 8.4 Summary of Changes:

In the sprint planning meeting with the product owner, the team reviewed the project brief in detail and broke down what the client wanted. The project scenario was also discussed, specifically the information and guidelines that were given about the cowboys (friendly agents) and aliens (unfriendly agents). Since this meeting, the team has performed individual preliminary research and literature reviews on DDM algorithms. Several additional project-oriented aspects were also considered, including what the user may wish to control/modify in their use cases, and ways in which scenario can be paramaterised. A potential software architecture described by 'Design Patterns', a popular framework for building extensible and maintainable object-oriented code, was also examined. In the week 3 team meeting individual work and research was consolidated, past experiences in related software fields were shared, and future actions for the sprint, including the initial report sections for the upcoming milestone, were allocated.

## 9.0 References

**Google.github.io. 2022. *Google Java Style Guide*. [online] Available at: <https://google.github.io/styleguide/javaguide.html#s2-source-file-basics> [Accessed 8 August 2022].**

**Refactoring Guru. 2022. Design Patterns. [online] Available at: <https://refactoring.guru/design-patterns> [Accessed 14 August 2022].**