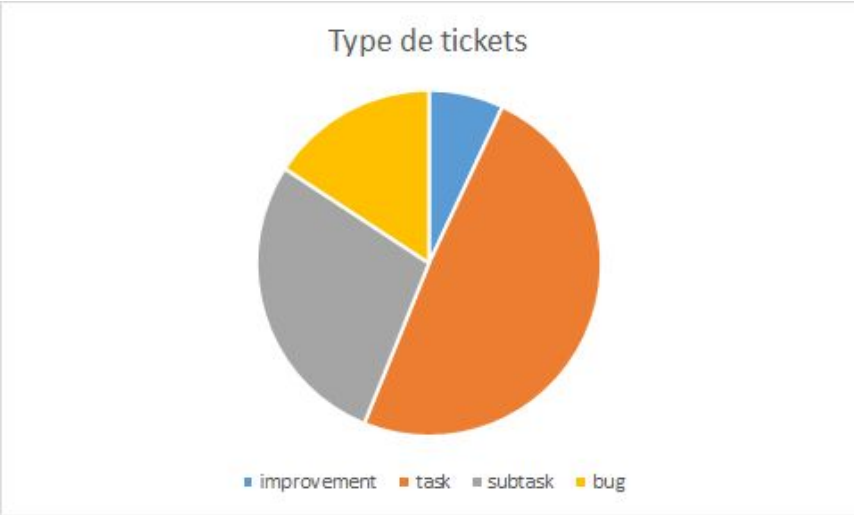
La Ligue Des Explorateurs : Code Review
Team IADC

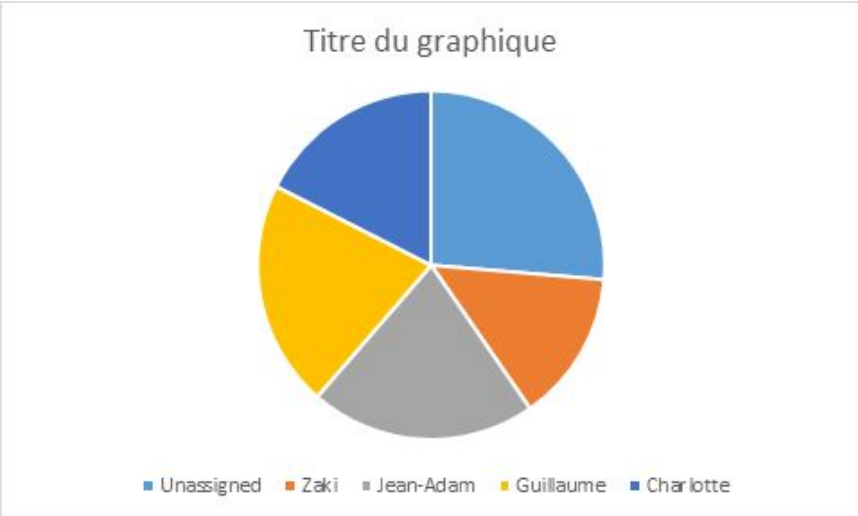
♦KANBAN :

Points positifs :

Premièrement, les tickets ont une bonne répartition en termes de catégories. Malgré le fait qu'une majorité soit des tâches, pas mal de tickets sont classés en bug, sous-tâche ou amélioration. Le groupe a donc pris soin de catégoriser leurs spécifications.



De plus, selon le Kanban, la répartition au sein du groupe des tâches est respectée puisque chaque membre a officiellement environ 20% des caractéristiques à leur nom. Néanmoins, 20% de tickets ne sont pas assignés. De plus, parmi ces tickets, certains sont pourtant catégorisé comme étant terminés.
Exemple : IADC-58



Points négatifs :

Le groupe possède 58 tickets, ce qui est plutôt moyen sur un projet de 6 mois à 4. De plus, on remarque que le nombre de tâches est supérieur au nombre de sous-tâches, ainsi, certaines tâches n'ont pas de sous tâches associées. En moyenne, chaque tâche a moins de 1 sous tâche associée. Ceci est un indicateur d'un découpage vertical pas assez fin. Peut-être les spécifications n'ont pas été découpées assez.
Par exemple le ticket (tâche): IADC-6 : Trouver l'île. Description : Différencier l'océan de la terre
Utiliser l'echo pour déterminer la distance à la terre ferme. Condition d'acceptation :
Ne jamais atterrir dans l'eau.

Pourrait être découpé ainsi : Le nom est correct, on peut très bien avoir pour tâche de trouver l'île et la localiser. Cependant, aucune sous-tâche n'est associée à cette tâche qui est plutôt complexe et épaisse. En outre, sa description laisse présager un certain découpage, mais qui n'est pas précisément effectué. Peut-être auraient-ils pu découper en plusieurs sous-tâches :
Sous tâche 1 : Être capable de différencier l'océan de la terre
Sous tâche 2 : Être capable de savoir à quelle distance le drone se trouve de la terre.

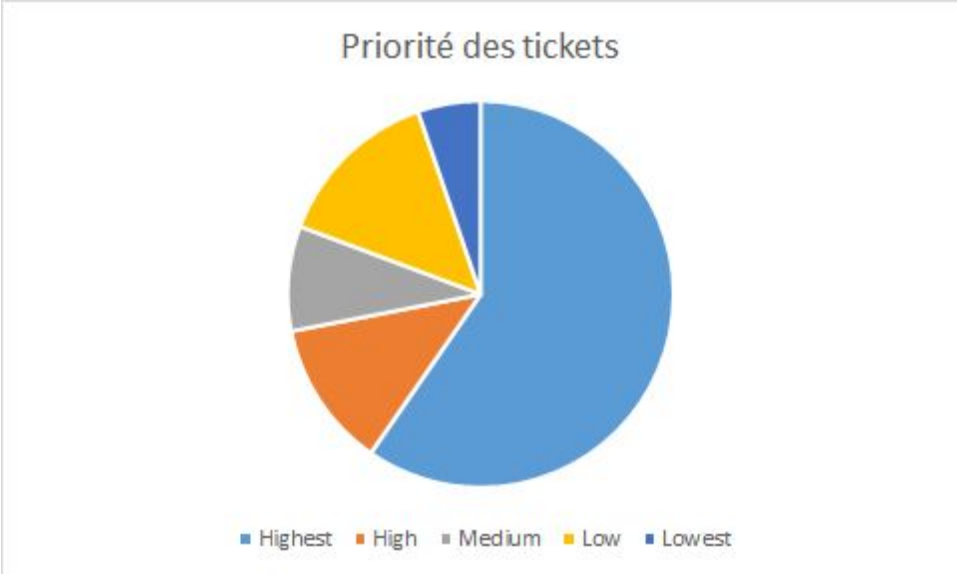
Conjointement, les tickets sont censés représenter une valeur client, être exprimés dans un langage que le client comprend. Ici, les spécifications sont trop souvent exprimées techniquement, en utilisant les noms des commandes du drone par exemple.

Exemple : IADC-31 : Choisir la méthode d'exploration. Description : Utiliser SCOUT, GLIMPSE et se déplacer [...]

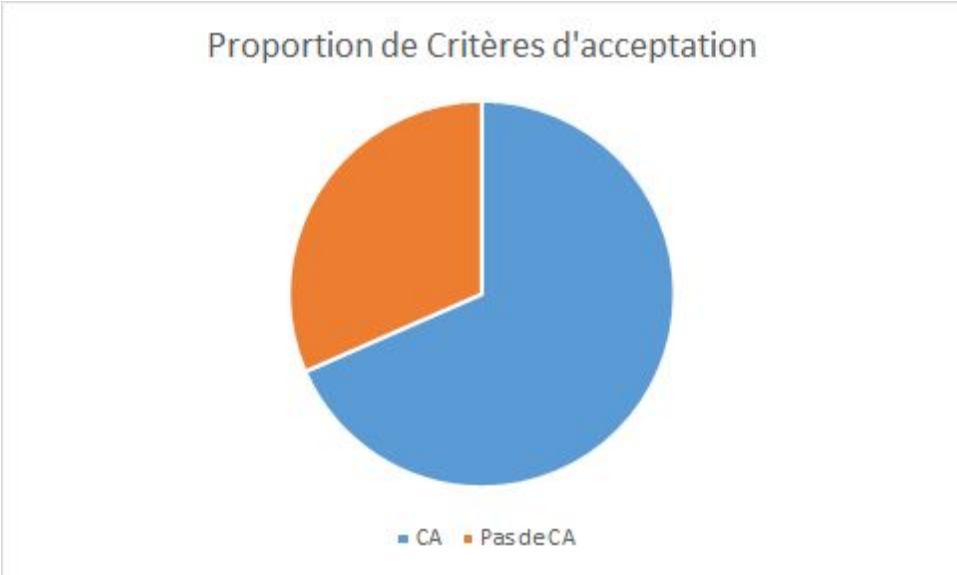
Pour pallier à cela, il aurait été judicieux de décrire ces spécifications autrement. Pour reprendre l'exemple précédent :

IADC-6 : Description : Utiliser l'Echo pour déterminer la distance à la terre ferme. ECHO est le nom d'une méthode agissant au sein du drone, le client ne sait pas ce que c'est. Peut-être aurait-il été plus judicieux de spécifier ceci de manière plus « client » par exemple : « Utiliser le Radar du drone afin de déterminer la distance à la terre ferme ».

Concernant la priorité des tickets, environ 60% des tickets sont classés en priorité « Highest ». Ceci ne paraît pas acceptable dans le sens où la priorité « Highest » correspond à une feature/bug à implémenter/résoudre en urgence afin d'assurer le bon fonctionnement du système ou la bonne santé du projet. Ainsi, environ un tiers des tickets se partagent les priorités « High », « Medium », « Low », « Lowest »



Enfin, environ un tiers des tickets ne comporte pas de critère d'acceptation. En effet, en analysant le Kanban, le client n'a aucun moyen de savoir à partir de quel point l'équipe de développeur juge la caractéristique comme terminée.



♦CODE :

Points positifs :

La structure du projet est clair : le découpage en package indique les trois grandes parties du projet chacune découper en trois sous-parties correspondant aux phases de jeu (aérienne, commune, terrestre).

La responsabilité entre les classes est bien répartie (quelques exceptions comme routine.aerial.ExploreRoutine qui aurait pu utiliser une sous routine "faire demi-tour" à l'image de

l'emploi de routine.aerial.TightUTourneRoutine qui permet de faire demi-tour après avoir parcourue toutes l'île dans un sens).

Le niveau important de documentation dans le projet permet de vite comprendre le code et le rôle de chaque élément.

Un très gros travail a également été effectué sur les tests, ils couvrent une grande partie du projet et sont très lisible, venant ainsi compléter la documentation pour une meilleure lisibilité du projet.

Points négatifs :

Les énumérations qui utilisent des données redondantes en paramètres comme le nom ou l'ordinal de celle-ci. Le projet n'est pas indépendant du format JSON utilisé.

Les tests sont nombreux et couvrent une grande partie du projet mais ils tests rarement les différents cas d'utilisation d'une méthode. Certaines méthodes peuvent même avoir un cas non couvert (exemple map.terrestrial.LandMap#setInterrestKey qui ne test pas le cas ou deux biomes ont le même score et sont donc départagés par leurs proximités à l'équipage. Les tests portant sur la stratégie comparent le résultat actuel à un résultat pré établie. Ce n'est pas très évolutif, si la stratégie change, les tests deviennent obsolètes. Dans la même optique, lorsqu'une décision est prise, les tests ne vérifient pas que celle-ci soit valide (exemple, un heading dans la direction actuelle du drone ou bien, une action dont le coût dépasse le budget disponible).

La répétition de nombreux switch case portant sur les directions à travers le code. On peut éviter cela en créant une méthode contenant un seul switch case dans map.common.Direction qui serait par la suite appelé à divers endroits du projet.