Table des matières

Présentation du mini-projet

Il s'agit de réaliser un logiciel permettant de **consulter des données météorologiques locales**.

On pilotera une station Maretron WSO-100 qui mesure la vitesse du vent, sa direction, la température de l'air, la pression atmosphérique et l'humidité relative. La mesure du vent est effectuée à l'aide de capteurs ultrason. L'absence de pièce en mouvement dans la station assure une longue durée de vie, une bonne fiabilité et un coût d'entretien nul.

La station Maretron WSO-100 émet périodiquement sur un bus CAN deux messages NMEA2000 (qui contiennent les mesures de la vitesse du vent, sa direction, la température de l'air, la pression atmosphérique et l'humidité relative).

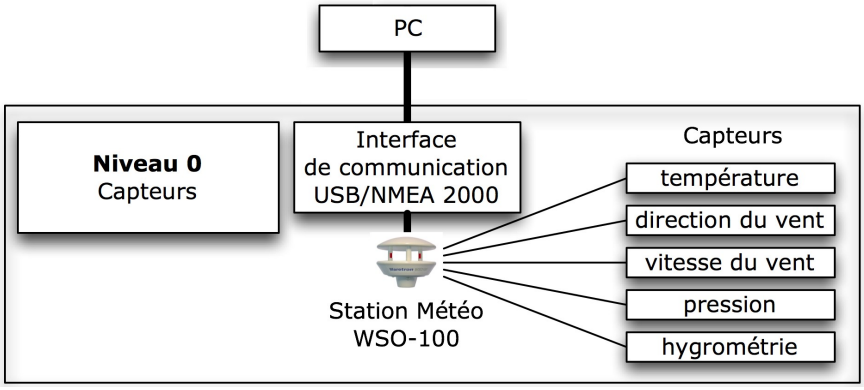
La station météo WSO-100, respectant les spécifications NMEA2000, nécessite une liaison bus CAN pour communiquer. Celle-ci est assurée par l'interface de communication PEAK CAN qui est elle-même reliée par une liaison USB au PC.

Remarque : consulter les annexes sur la norme NMEA2000 et la station Maretron WSO-100

Ce mini-projet sera développé en équipe de 4 étudiants **afin de mettre en application l'utilisation de Subversion (un logiciel de gestion de versions)**.

Expression du besoin

Dans le cadre d'un système d'informations météorologiques locales, on a besoin d'un logiciel capable d'afficher à l'écran les mesures de vitesse du vent et sa direction, de température de l'air, de pression atmosphérique et d'humidité relative.



<u>Exemple d'affichage de la version actuelle :</u>

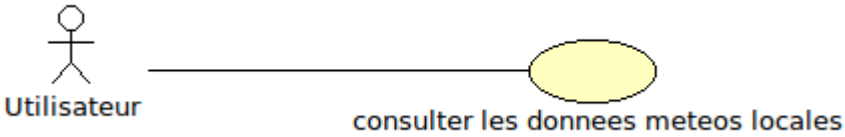
```
$ ./station-meteo
Station Météo à Avignon

Temperature : 28 °C
```

Remarque : les mesures doivent être absolument horodatées lors de leur affichage.

Moyens préliminaires disponibles et contraintes de réalisation

Spécifications

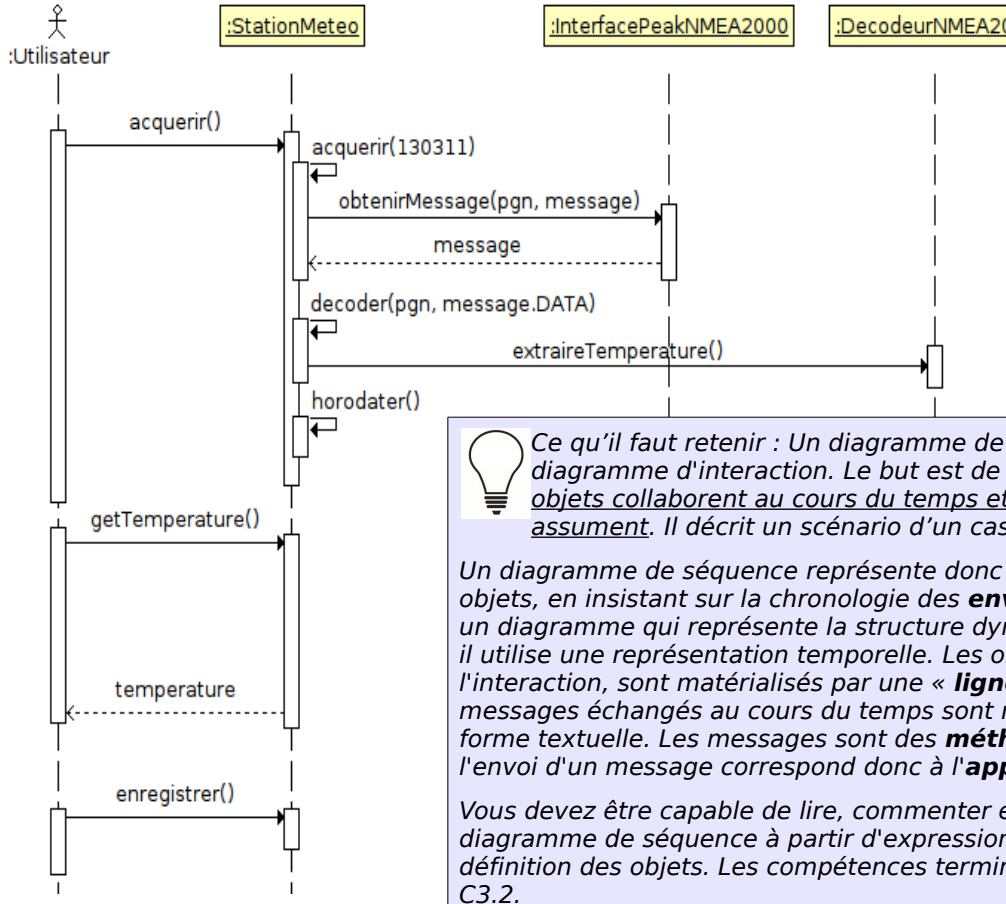
Le **diagramme des cas d'utilisation** (*Use Case*) du système est le suivant :



Ce qu'il faut retenir : Un diagramme des cas d'utilisation décrit les fonctionnalités attendues du système du point de vue d'un utilisateur. Les Cas d'Utilisation (CU) recentrent l'expression des besoins sur les utilisateurs. Les cas d'utilisation sont donc très utiles pour représenter ce que doit faire un système par rapport à son environnement.

Vous devez être capable, vis à vis d'un diagramme de cas d'utilisation, de le lire, le commenter et l'expliquer au regard des fonctionnalités décrites dans le cahier des charges. Vous devez pouvoir aussi le modifier et le compléter localement. Les compétences terminales visées sont : C3.1 et C3.2.

Pour vous aider, on vous fournit le **diagramme de séquence** « consulter les données météos locales » dans sa version actuelle :



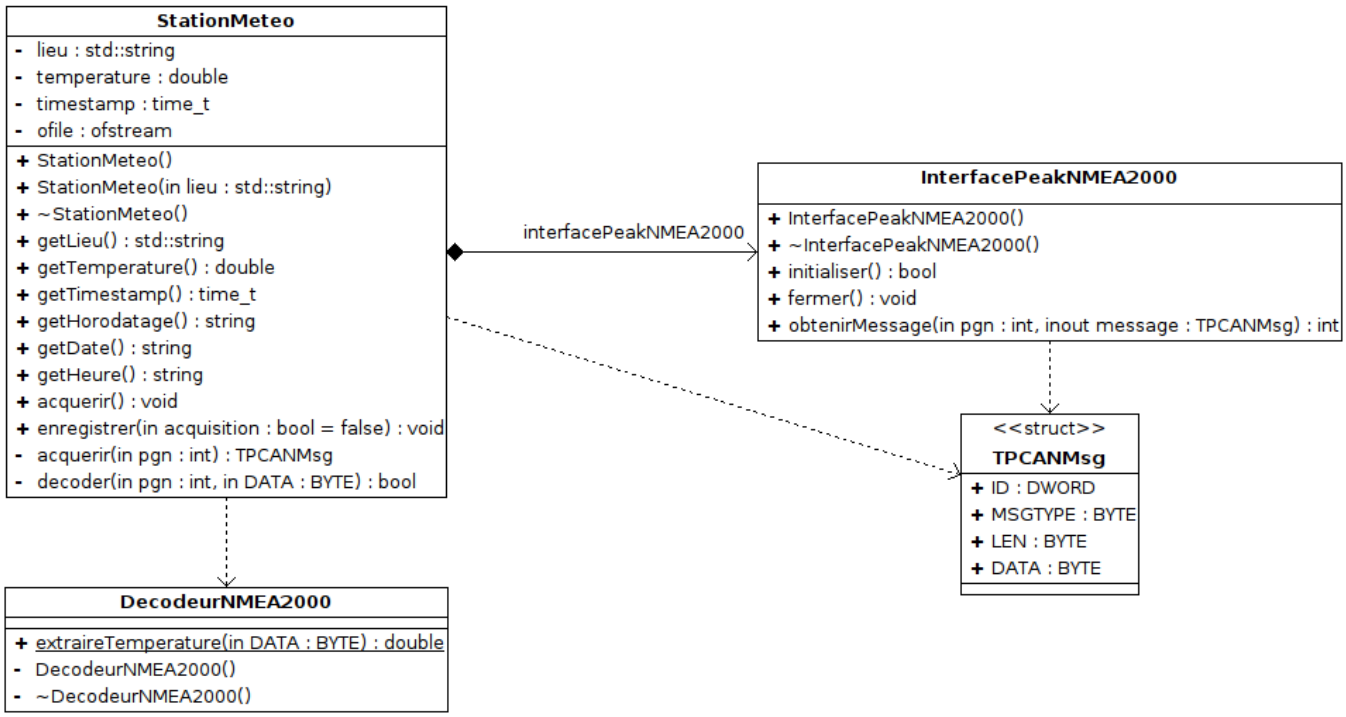
Ce qu'il faut retenir : Un diagramme de séquence est un diagramme d'interaction. Le but est de décrire comment les objets collaborent au cours du temps et quelles responsabilités ils assument. Il décrit un scénario d'un cas d'utilisation.

*Un diagramme de séquence représente donc les interactions entre objets, en insistant sur la chronologie des **envois de message**. C'est un diagramme qui représente la structure dynamique d'un système car il utilise une représentation temporelle. Les objets, intervenant dans l'interaction, sont matérialisés par une « **ligne de vie** », et les messages échangés au cours du temps sont mentionnés sous une forme textuelle. Les messages sont des **méthodes d'une classe** et l'envoi d'un message correspond donc à l'**appel de cette méthode**.*

Vous devez être capable de lire, commenter et compléter un diagramme de séquence à partir d'expressions textuelles et / ou de la définition des objets. Les compétences terminales visées sont : C3.1 et C3.2.

Vous devez aussi être capable de produire du code à partir d'un diagramme de séquence.

Le **diagramme de classes** de la version actuelle est le suivant :



Contrainte de développement

On appliquera un **cycle de développement itératif et incrémental**.

Ce mini-projet sera développé en deux itérations contenant les fonctionnalités suivantes :

- Version 1.0 :
  - récupérer et afficher les mesures de la vitesse du vent, sa direction, la température de l'air, la pression atmosphérique et l'humidité relative
  - horodater les mesures et l'afficher (date et heure)

- Version 2.0 :
  - enregistrer les mesures horodatées dans un fichier de type CSV
  - valider les mesures avant affichage (cf. domaine de définition)

Ce qu'il faut retenir : Les diagrammes de classes (et d'objets) représentent la structure statique d'un système : les classes, (les objets,) leurs structures internes (attributs et méthodes) et leurs relations.

Vous devez être capable de lire, commenter et compléter un diagramme de classes en s'appuyant sur les dossiers de spécification, de conception préliminaire et les documentations techniques. La compétence terminale visée est alors : C3.9.

Vous devez aussi être capable de d'identifier et d'interpréter les éléments pertinents d'un diagramme de classes de manière à pouvoir traduire sous la forme de code orienté objet les résultats de la conception détaillée. Par exemple, vous codez tout ou partie d'une méthode. La compétence terminale visée est alors : C4.4.

Contrainte de l'environnement

Système d'exploitation : choix libre (Linux ou Windows)
Environnement de développement : choix libre
Atelier de génie logiciel : bouml
Logiciel de gestion de versions : **subversion**

Contrainte économique

Aucune

Documents et moyens technologiques mis à disposition

Documents :

- des tutoriels et documentations sur Subversion (dans le répertoire doc)

Ref.	Description
tutoriel-riouxSVN.odt tutoriel-riouxSVN.pdf	tutoriel au format ODT et PDF présentant la mise en œuvre de subversion chez l'hébergeur spécialisé **RiouxSVN** (http://riouxsvn.com/)
svn-refcard.pdf	le « *Subversion Quick Reference Card* » contenant une description de l'ensemble des sous commandes et options de la commande **svn**.
imerir/	Dossier contenant un ensemble de cocuments au format PDF sur Subversion réalisés par **Michael Jégat** (Corexpert) pour l'école d'ingénieur IMERIR
svn-exemple-pratique.odt svn-exemple-pratique.pdf	document au format ODT et PDF présentant un exemple simple et détaillé de l'utilisation de subversion par deux développeurs

- cours et exemples sur le gestion de fichiers en C++ (`cours-c-c++-fichiers.pdf` et `exemples-fichier.zip`)
- une annexe sur les fichiers de gestion de projet (`Changelog`, `TODO` et `README`)

Moyens technologiques :

- Accès Internet
- vidéo-projecteur avec écran interactif et rétro-projecteur

Exigences qualité à respecter

Exigences qualité sur le produit à réaliser

Le produit à réaliser doit répondre aux facteurs de qualité suivants :

- maniable : il sera facile d'emploi avec une interface homme-machine simple et conviviale
- robuste : il conservera un fonctionnement conforme aux spécifications après un arrêt
- normal ou d'urgence; garantira la validité des informations échangées.
- maintenable : il offrira une bonne facilité de localisation et correction des problèmes résiduels.
- adaptabilité : il facilitera la suppression, l'évolution de fonctionnalités existantes ou l'ajout de nouvelles fonctionnalités
- portabilité : il minimisera les répercussions d'un changement d'environnement logiciel et matériel

Exigences qualité sur le développement

En ce qui concerne les exigences qualité du développement :

- La modélisation UML doit être réalisée avec un atelier de génie logiciel (bouml)
- L'architecture du logiciel sera Orientée objet.
- Le codage doit respecter le standard C/C++ en cours dans la section
- Un utilitaire de compilation automatisé de type « make » sera utilisé
- Le gestionnaire de gestion de versions utilisé sera **subversion**

Exigences qualité sur la documentation à produire

Les exigences qualité à respecter, relativement aux documents, sont :

- sur leur forme : respect de normes et de standards de représentation, homogénéité, lisibilité, maintenabilité ;
- sur leur fond : complétude, cohérence, précision.

Exigences qualité sur la livraison

Les produits à mettre à disposition du client sont :

- la documentation (fichiers `Changelog`, `TODO`, `README` et `configuration.txt`) ;
- les codes sources de l'application de la dernière version livrable, ainsi que le fichier de type Makefile.

Ces produits seront livrés sous forme informatique regroupés dans une archive au format `tar.gz` ou `zip`. Le nom de l'archive sera formaté de la manière suivante : **`mp1b-teamN-vX.Y.tar.gz`** ou **`mp1b-teamN-vX.Y.zip`** où :

- N représente le numéro de l'équipe de développement (numéro donné par l'enseignant)
- X le numéro de version majeur de l'application
- Y le numéro de version mineur de l'application (0 par défaut)

Exigences qualité sur l'environnement d'exploitation

Aucune

Exploitation pédagogique

En général dans les activités de mini-projet :

Activités professionnelles	Coopération et communication
Capacité	C1 COMMUNIQUER
Compétences terminales susceptibles d'être abordées et évaluées	
Compétence terminale	Critères d'évaluation
C1.3 Travailler en équipe	- Respect des autres membres de l'équipe. Compréhension du fonctionnement du travail en groupe. - Régularité et pertinence des informations remontées au responsable. - Clarté de la synthèse.
C1.6 Présenter la mise en oeuvre d'une solution informatique	- Qualité de la présentation. Maîtrise du fonctionnement de la solution. - Respect du cahier des charges.

Activités professionnelles	Gestion de projet
Capacité	C2 ORGANISER
Compétences terminales susceptibles d'être abordées et évaluées	
Compétence terminale	Critères d'évaluation
C2.1 S'intégrer dans une équipe de projet	- Identification des acteurs du projet. Identification des rôles respectifs de chacun. - Identification des ressources matérielles et logicielles utilisées par les membres de l'équipe de projet. - Respect des répartitions des tâches. Respect des répartitions des responsabilités partagées.
C2.2 Structurer son intervention dans une démarche de projet	- Consignation de l'état d'avancement de l'intervention. - Respect des délais. Consignation des dépassements de délais.
C2.3 Intervenir dans la gestion de projet	- Identification des indicateurs. - Renseignement des indicateurs de projet. - Repérage des écarts. - Pertinence des adaptations proposées.
C2.4 Prévenir les risques d'échec dans la mise en oeuvre d'une solution au cours d'un projet	- Fréquence des sauvegardes des versions successives des logiciels. Mise à jour des dossiers associés aux logiciels. - Qualité de la mise à jour de la documentation. - Identification des risques. - Pertinence du moyen d'action proposé.

Activités professionnelles	Analyse et spécification d'un système informatique à développer Conception générale et détaillée
Capacité	C3 CONCEVOIR
Compétences terminales susceptibles d'être abordées et évaluées	
Compétence terminale	Critères d'évaluation
C3.1 Analyser un dossier de spécification	- Repérage des liens entre les spécifications fournies et le besoin. - Critères retenus pour caractériser les fonctionnalités.
C3.2 Définir l'architecture globale d'un prototype ou d'un système	- Qualité de la transcription des spécifications en pseudo-code ou sous forme d'algorigramme. - Compréhension des critères de choix. - Définition du rôle de chacun des moyens matériels et logiciels. - Qualité du document fourni qui doit faire clairement apparaître les choix matériels et logiciels.

Activités professionnelles	Codage et réalisation Intégration et interconnexion de systèmes
Capacité	C4 REALISER
Compétences terminales susceptibles d'être abordées et évaluées	
Compétence terminale	Critères d'évaluation
C4.8 Coder un module logiciel	- Qualité de la traduction en code de la spécification logicielle. Respect de la syntaxe du langage utilisé. - Degré de maîtrise de la manipulation des pré-processeurs, compilateurs, interpréteurs, assembleurs, lieurs, ... - Choix de la méthode de mise au point. Degré de maîtrise des outils de simulation et de mise au point. - Utilisation correcte d'un système de fichiers. - Qualité du document. Pertinence des commentaires. Clarté de la présentation des points d'entrée du module.
C4.9 Intégrer un module logiciel dans une application	- Pertinence du choix des interfaces sélectionnées. Qualité de l'appel des méthodes. - Pertinence des modifications du module (dérivation, association ou agrégation de classes).

Plus spécifiquement dans ce mini-projet :

Activités professionnelles	Installation, exploitation, optimisation et maintenance
Capacité	C5 INSTALLER
Compétences terminales susceptibles d'être abordées et évaluées	
Compétence terminale	Critères d'évaluation
C5.7 Mettre en oeuvre un environnement de programmation - Installer et configurer un environnement de développement. **- Exploiter les outils de travail collaboratif.** - Utiliser les assistants de l'environnement pour créer les squelettes de l'application, des classes, des fonctions et des messages.	- Fonctionnement de l'environnement de développement. - Verrouillage des fichiers. Historique des modifications. Gestion des droits des utilisateurs. - Propriétés du squelette de chacun des éléments.

Planification des tâches spécifiques au mini-projet

Ref.	Description	L	M	M	J	V	S	D	L	M	M	J	V
T2.1	Interprétation des spécifications logicielles et identification des fonctions principales												
T2.3	Conception de l'architecture des interfaces homme - machine												
T2.8	Utilisation des diagrammes de classe pour produire une maquette de l'application												
T3.3	Codage et assemblage des modules logiciels (dans le respect des standards entreprise)												
T3.4	Fabrication de modules logiciels réutilisables et réalisation de la documentation associée												
T3.7	Élaboration de documents de suivi de réalisation et de codage												
T8.1	Intégration et travail dans une organisation par projet												
T8.5	Renseignement des indicateurs permettant le suivi d'un projet												
T8.7	**Gestion des évolutions des versions successives des logiciels et des documents produits**												
T9.1	Intégration et travail en équipe												
T9.2	Exposé et argumentation des choix de conception, des choix techniques et des résultats de travaux auprès du service concerné ou du client (communication écrite et orale)												

Travail à réaliser

Étape n°1 : communiquer, organiser et planifier

Après avoir formé des équipes de 4 personnes, la première étape va consister à désigner un chef de projet. Cette personne sera responsable de la gestion de projet et servira d'intermédiaire avec le client. Une fois le chef de projet désigné, vous devez réfléchir au projet, aux fonctionnalités et aux choix techniques (structure de données, algorithmes, ...).

Vous devez aussi définir une organisation de travail (dépôt Subversion, gestion des *bugs*, convention de nommage, ...). Enfin, vous devez planifier les tâches et les attribuer aux membres de l'équipe de projet.

Toutes ces informations devront être écrites sous format informatique (les fichiers `Changelog`, `TODO` et `README`) et devront être disponibles à chaque membre du projet.

A la fin du temps attribué à cette étape, le chef de projet présentera brièvement son plan d'action et des questions pourront être posées au client concernant les choix techniques, le besoin, etc.

Production attendue à la fin de cette étape :

- Contenu minimal du fichier `README` à la fin de cette étape :
  - Nom du logiciel :
  - Date de début du mini-projet :
  - Objectif :
  - Équipe de développement : nom, prénom et <adresse courriel> de chaque membre
  - Ce que le logiciel doit faire : description détaillée des fonctionnalités offertes

Au fur et à mesure de l'avancement du mini-projet, vous ajouterez les informations suivantes :

  - Numéro de version du logiciel :
  - Date de cette version du logiciel :
  - Ce que le logiciel fait dans cette version :
  - Défauts constatés non corrigés :

- Contenu minimal du fichier `TODO` à la fin de cette étape : la liste des tâches et leur attribution

- Contenu minimal du fichier `Changelog` à la fin de cette étape : aucun

Étape n°2 : préparer et installer

Vous devez préalablement installer et préparer votre environnement de développement (éditeur et chaîne de fabrication `make`/`g++`).

Pour la mise en œuvre du logiciel de gestion de versions (subversion), vous devez suivre le tutoriel fourni. Une fois les comptes créés, vous devez les communiquer à l'administrateur.

Vous devez assurer une gestion de configuration de votre mini-projet. Pour cela, vous allez créer un fichier `configuration.txt` contenant pour chaque membre de l'équipe l'identification de votre plate-forme de développement : indiquer précisément le nom et la version de chaque outil utilisé ainsi que votre système d'exploitation.

Le chef de projet pourra alors mettre à jour dans votre dépôt subversion les fichiers `Changelog`, `TODO` et `README` créés à l'étape n°1 et le fichier `configuration.txt`.

Production attendue à la fin de cette étape :

- la chaîne de fabrication est fonctionnelle
- le dépôt subversion dédié à votre mini-projet est utilisable par chaque membre de l'équipe
- le dépôt subversion dédié à votre mini-projet est à jour (fichiers `Changelog`, `TODO`, `README` et `configuration.txt`)

Étape n°3 : concevoir et réaliser

Cette étape va consister à réaliser le projet en suivant un développement itératif et incrémental.

Pour l'utilisation du logiciel de gestion de versions (subversion), vous devez vous référer au tutoriel fourni.

Lorsqu'une version est terminée, vous devez contacter le client pour qu'il valide le logiciel. Si des défauts (*bugs*) sont trouvés lors de l'utilisation, ils devront être corrigés dans les plus brefs délais. Si plus aucun défaut (*bug*) n'est trouvé, la version sera acceptée définitivement et la version suivante pourra être testée.

Ainsi, une version possède trois états :

- en développement : la version est en cours de développement (***trunk***).
- livrée : la version est livrée chez le client qui l'utilise (***tags***).
- acceptée : le client a utilisé la version livrée et n'a plus détecté de problèmes d'utilisation. Lorsqu'une version est acceptée, la version suivante peut être livrée si elle est terminée.

Il vous faudra maintenir continuellement les fichiers `Changelog`, `TODO` et `README` (ces fichiers faisant partie d'une version livrable au client). La description de ces fichiers est fournie en Annexe.

Production attendue à la fin de cette étape :

- un version livrable a été au minimum produite par l'équipe de développement
- le dépôt subversion dédié à votre mini-projet est à jour (fichiers sources, `Changelog`, `TODO`, `README` et `configuration.txt`)

Étape n°4 : présenter oralement

Vous présenterez oralement :

- le diagramme de classes final
- les fonctionnalités validées
- les fonctionnalités non implémentées
- les défauts constatés non corrigés
- une démonstration de l'application

Grille d'évaluation

Mini-projet :		Livraison finale		
Équipe n°__ :		Version : ____		
Critères		-	0	+
Gestion de projet	Respect de la méthodologie de développement itérative			
	Respect du travail en équipe			
	Respect du cahier des charges			
	Respect et suivi de la planification			
	Les fichiers README, TODO et Changelog existent et sont correctement renseignés			
	Les fichiers (sources, TODO, …) sont accessibles et à jour sur le serveur **subversion**			
Oral	Qualité de la présentation, précision, rigueur, clarté			
	Utilisation des moyens mis à la disposition et du vidéo-projecteur			
Démonstration	Qualité de la démonstration orale : précision, rigueur, clarté, ...			
Application	L'application livrée respecte les contraintes			
	L'application livrée respecte les exigences qualité			
Entretien	Capacité à répondre avec pertinence, précision et exactitude			
	Capacité à rechercher et à exploiter une documentation			
	Capacité à argumenter et à réagir aux objections			
Bilan	État d'avancement			
Remarques :				

Annexe : les fichiers de gestion de projet

Changelog

Un ChangeLog, littéralement **Journal des modifications** (en anglais), désigne souvent un fichier qui contient l'énumération de ce que les personnes collaborant à un projet ont effectué comme travail sur ce dernier. Ce fichier est souvent un simple fichier texte, assez brut, avec éventuellement des sections correspondant aux différentes sous-parties du projet. On peut également y trouver des noms de personnes qui ont réalisé ces tâches. Ce terme est directement issu du monde des développeurs de logiciels, notamment celui des développeurs de logiciels libres, afin que tout le monde puisse savoir dans quelle direction le projet a évolué à travers le temps, quelle est sa vitalité (s'il avance beaucoup et vite, s'il est en plein essor ou abandonné depuis un an, ou seulement en phase de corrections de bugs). Dans tous les cas, cela constitue également une invitation à contribuer en sachant quelles sont les dernières évolutions, ce qu'il y a à tester...

TODO

Une *todo list* (ou **une liste des choses à faire**) est un procédé simple et efficace qui permet de se concentrer sur une tâche d'un projet sans pour autant perdre de vue les autres tâches à accomplir. Les listes à faire se déclinent de multiples façons : par exemple, un chef de projet qui note les bogues à corriger et les fonctionnalités à programmer construit une *todo list*. Plus trivialement, un post-it avec une liste de courses à faire est aussi une *todo list*.

Une *todo list* désigne parfois un fichier qui contient l'énumération de ce que les personnes se fixent comme tâches à réaliser. Ce fichier est souvent un simple fichier texte, assez brut, avec éventuellement des sections correspondant à différents domaines d'action ou aux différentes sous-parties d'un projet.

Pour le travail en équipe, on peut également y trouver des noms de personnes à qui ces tâches échoient.

Les éléments présents dans une *todo list* sont généralement biffés une fois réalisés, afin de mesurer rapidement l'avancement global. Pour des projets logiciels, ces éléments devraient à terme se retrouver dans le journal des améliorations et modifications apportées au programme ChangeLog.

README

Un fichier readme (**lisezmoi** en français) est, en informatique, un fichier contenant des informations sur les autres fichiers d'un répertoire.

Un tel fichier est généralement un fichier texte appelé « README.TXT », « README.1ST », « READ.ME » ou simplement « **README** », parfois localisé dans les distributions françaises en « LISEZMOI.TXT », etc. Son contenu varie mais inclut d'ordinaire des instructions d'exploitation, une liste des noms et utilités des autres fichiers, des informations sur la personne les ayant créés, la version de l'application, sa description, voire la licence applicable.

Annexe : la norme NMEA2000

(NMEA = National Marine Electronics Association)

Le NMEA 2000 est un standard d'octobre 2001 pour les réseaux Serial-Data à 250 kbits/s (sur 200 m) utilisant un contrôleur CAN en mode étendu et comprenant 50 noeuds.

Les messages NMEA 2000 sont organisés en PG (Parameter Group) qui sont identifés par un PGN (Parameter Group Number) qui apparairaît dans le champ ID (Identifier) d'un message CAN.

Structure des identifiants étendus (29 bits) des messages NMEA 2000 :

28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Priority			Libre		MSB ID (pgn)								LSB ID (pgn)								Source Address *							
			■	■																								

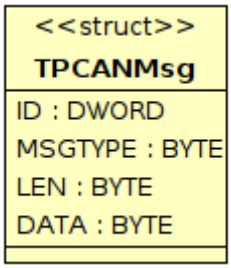
Remarque sur le champ Source Address :

Chaque appareil du réseau doit posséder une adresse source sur 8 bits (*). Les adresses 0 à 251 (soit un total de 252 adresses disponibles) sont réservées à cet usage. L'adresse 255 est une adresse globale qui spécifie un envoi à tous les noeuds du réseau (*broadcast*). La valeur 254 est une adresse nulle réservée pour rapporter un problème lorsqu'une adresse n'a pu être trouvé. Les adresses 253 et 252 ont été réservées à un usage futur.

Structure d'un message CAN NMEA2000 :

Il existe deux types de messages CAN : standard ou étendu. Un message CAN contient les champs suivants :

- Identifiant : sur 11 bits en standard ou sur 29 bits en étendu
- Longueur : nombre d'octets de données dans le message (de 0 à 8 octets max)
- Données : 0 à 8 octets



Le standard **NMEA2000** définit un message CAN avec les contraintes suivantes :

- **Identifiant : sur 29 bits et contient le PGN**
- **Longueur : 8**
- **Données : 8 octets**

Annexe : la station Maretron WSO-100

La station météo Maretron WSO100 mesure la vitesse et la direction, température de l'air, la pression barométrique et l'humidité relative. La mesure du vent est réalisée en utilisant des capteurs à ultrasons, ce qui signifie qu'il n'y a pas de pièces mobiles susceptibles de s'user ou de se coincer.

NMEA 2000® Parameter Group Numbers (PGNs)

Description	PGN #	PGN Name	Default Rate
Periodic Data PGNs	130306	Wind Data	10 Times/Second
	130310	Environmental Parameters	10 Times/Second
	130311	Environmental Parameters	2 Times/Second
Response to Requested PGNs	126464	PGN List (Transmit and Receive)	N/A
	126996	Product Information	N/A
	126998	Configuration Information	N/A
Protocol PGNs	059392	ISO Acknowledge	N/A
	059904	ISO Request	N/A
	060416	ISO Transport Protocol, Connection Management	N/A
	060160	ISO Transport Protocol, Data Transfer	N/A
	060928	ISO Address Claim	N/A
	065240	ISO Address Command	N/A
	126208	NMEA Complex Request/Command/Ack.	N/A
Maretron Proprietary PGNs	126720	Configuration	N/A

PGN 130306 – Wind Data

The WSO100 uses this PGN to provide a regular transmission of apparent wind speed and direction data. The factory default for periodic transmission rate is ten times per second. The transmission of this PGN can be disabled (see PGN 126208 – NMEA Request Group Function – Transmission Periodic Rate).

Field 1: SID – The sequence identifier field is used to tie related PGNs together. For example, the WSO100 will transmit identical SIDs for 130306 (Wind Data) and 130311 (Environmental Parameters) to indicate that the readings are linked together (i.e., the data from each PGN was taken at the same time although they are reported at slightly different times).

2: Wind Speed – This field is used to indicate the wind speed in units of 10mm/second.

3: Wind Direction – This field is used to indicate the wind direction in units of 0.0001 radians/second.

4: Wind Reference – This field is set to a value of 0x02 to indicate that the wind reading is an apparent wind speed and direction.

5: Reserved (21 bits) – This field is reserved by NMEA; therefore, this field always contains a value of 0x1FFFFF (the WSO100 sets all reserved bits to a logic 1)

PGN 130311 – Environmental Parameters

The WSO100 uses this PGN to provide a regular transmission of outside temperature, relative humidity, and atmospheric pressure. The factory default for periodic transmission rate is twice per second. The transmission of this PGN can be disabled (see PGN 126208 – NMEA Request Group Function – Transmission Periodic Rate).

Field 1: SID – The sequence identifier field is used to tie related PGNs together. For example, the WSO100 will transmit identical SIDs for 130306 (Wind Data) and 130311 (Environmental Parameters) to indicate that the readings are linked together (i.e., the data from each PGN was taken at the same time although they are reported at slightly different times).

2: Temperature Instance – The WSO100 sets this field to a value of 0x01 to indicate that the temperature reading is a reading of outside temperature.

3: Humidity Instance – The WSO100 sets this field to a value of 0x01 to indicate that the relative humidity reading is a reading of outside humidity.

4: Temperature – This field is used to indicate the outside air temperature in units of 0.01°K.

5: Humidity – This field is used to indicate the relative humidity in units of 0.004%.

6: Atmospheric Pressure – This field is used to indicate the barometric pressure in units of 100 Pa.

