Conception d'un système interactif en Java

MVC en MVC : Mitigeur Vitesse-Cap selon l'architecture Modèle-Vue-Contrôleur *(1ere partie)*

Mathieu Petit, Meriam Horchani

19 mai 2008

Résumé

Par rapport à d'autres langages, Java est livré avec des bibliothèques graphiques qui permettent l'implémentation de systèmes interactifs riches et performants. Deux d'entre elles sont plus communément utilisées : AWT (Abstract Window Toolkit) et son évolution multi-plateforme SWING. Cette bibliothèque structure tous ces composants selon le patron Modèle-Vue-Controleur, inventé dans les années 70 avec le language SmallTalk. Vous verrez dans ce TD quels sont ses avantages pour la construction de systèmes interactifs.

Cette première partie sera dédiée à la conception centrée utilisateur : vous partirez d'un échange entre des concepteurs et des utilisateurs pour construire l'arbre des tâches de l'application. Puis vous implémenterez en Java les premiers prototypes en respectant l'architecture MVC et le principe d'identification de la tâche.

Table des matières

1 Du scénario vers la tâche et vers les concepts

Le point de départ de la conception centrée utilisateur est donnée par une description de la tâche et des besoins, qui doit être établie en confrontant les concepteurs et les utilisateurs finaux du système. À partir de cette analyse, il est possible d'écrire un scénario d'utilisation du système à concevoir, puis d'en tirer les concepts-clés et la décomposition de la tâche décrite. Nous détaillons chaque étape.

1.1 Échange avec les utilisateurs

Pour le système de poste de commande numérique qui nous concerne, voici une transcription textuelle de l'analyse de la tâche menée par deux concepteurs avec deux futurs utilisateurs :

CONCEPTEUR 1: Alors voilà, nous sommes réunis pour discuter de la future plateforme numérique de navigation.

MARIN 1: La quoi ?

CONCEPTEUR 2: On s'est dit que ça pourrait être bien de vous faire participer à une conception centrée utilisateur pour l'analyse Tâche-Concepts de ce système interactif ...

MARIN 2: ... Excuse moi là, mais moi et mon collègue on comprend pas trop où tu veux en venir avec ton concept de plateforme tâche-navigation.

MARIN 1: Un peu de clarté.

CONCEPTEUR 2: Oui, bon ! En fait, ce n'est pas très compliqué. Vous savez sans doute que les nouveaux postes de navigation seront équipés en matériel numérique ...

CONCEPTEUR 1: ... des ordinateurs .. avec écran, clavier et souris.

CONCEPTEUR 2: Voila, et il s'agit pour nous de concevoir le meilleur système de navigation pour que vous puissiez l'utiliser de la façon la plus simple possible.

MARIN 1 & 2: Haaaa, d'accord.

CONCEPTEUR 1: Pour cela, nous aimerions savoir en quoi consiste votre boulot de navigateur, quelles sont les opérations habituelles de conduite du bateau.

MARIN 1: Oui, c'est déjà plus concret. Donc vous dites que notre poste de navigation va être remplacé par un ordinateur ?

CONCEPTEUR 1: C'est ça, vous utiliserez un ordinateur pour contrôler tout les paramètres du bateau.

CONCEPTEUR 2: Et nous nous travaillons sur les opérations de navigation.

CONCEPTEUR 1: Alors disons que vous êtes en mer, le bateau est à l'arrêt et votre but est de rejoindre le port le plus proche. Comment faîtes-vous ?

MARIN 1: C'est plutôt simple, vous savez. Sur une carte papier, je commence par déterminer le cap à prendre pour rejoindre le port ...

MARIN 2: Moi je commence par dire à l'opérateur machine de faire tourner le moteur à vide.

MARIN 1: C'est vrai, je fais ça aussi. Et après je règle le cap sur le poste de navigation...

CONCEPTEUR 1: Dans l'ordre : 1) Mise en route des moteurs, 2) Choix du cap, 3) Réglage du cap

MARIN 2: Oui, c'est ça.

CONCEPTEUR 2: Et c'est tout ?

MARIN 2: Bien non, après on va choisir et régler la vitesse sur le poste de navigation. Sinon le bateau n'avance pas.

CONCEPTEUR 2: Bien sûr. Après cela, le bateau va partir vers le bon cap et à la bonne vitesse ?

MARIN 1: Oui, comme prévu. C'est assez simple.

CONCEPTEUR 2: Et si dans le chemin vers le port, un bateau vous coupe la route ?

MARIN 1: Dans ce cas nous avons le choix, personnellement, je réduis ma vitesse ...

MARIN 2: ... Moi par contre, je modifie le cap pour effectuer une manoeuvre d'évitement.

CONCEPTEUR 1: Si je comprends bien, vous dites qu'à tout moment vous pouvez jouer soit sur le cap, soit sur la vitesse ?

MARIN 1: Oui, bien sûr. C'est normal, non ?

CONCEPTEUR 2: C'est qu'avec votre description du démarrage du bateau, on pouvait croire qu'il fallait impérativement régler le cap avant le vitesse.

MARIN 1: C'est-à-dire que l'on pourrait aussi régler d'abord la vitesse.

MARIN 2: Oui, rien ne l'empêche, mais ce serait assez étrange ... il faut bien savoir où l'on va avant d'y aller ...

MARIN 1: En fait, notre poste de commande est divisée en deux parties, l'une pour le cap et l'autre pour la vitesse, avec à chaque fois 2 boutons qui nous permettent soit d'incrémenter, soit de décrémenter le cap ou la vitesse.

CONCEPTEUR 2: Vous avez 4 boutons devant vous ?

MARIN 2: Voilà, et il n'y a aucun ordre particulier pour les utiliser, on peut même agir en même temps sur le cap et la vitesse.

CONCEPTEUR 1: Et vous pouvez toujours utiliser le bouton "+" pour l'accélération ? J'imagine que vous ne pouvez pas accélérer indéfiniment ?

MARIN 1: Non, évidement, mais on le voit bien quand le bateau n'accélère plus.

CONCEPTEUR 1: Mais vous pouvez toujours appuyer sur le bouton ?

MARIN 1: Oui, mais cela ne sert a rien. De toute façon le bateau ne va pas au delà de 25 noeuds.

CONCEPTEUR 1: Messieurs, il semble que nous ayons fait le tour. C'est effectivement un système assez simple et je pense que nous n'aurons pas de difficultés supplémentaires pour la conception. Merci de votre participation et à bientôt pour les tests.

Question 1 *À partir de cet échange, écrivez en quelques lignes, un scénario décrivant ce que fait le marin à son poste de navigation pour diriger son bateau. Y a-t-il une partie de la tâche qui n'a pas été exprimée par les marins ? Si oui, quelle à été la responsabilité des concepteurs lors de l'échange ?*

1.2 Scénario nominal

Nous proposons pour la suite de considérer le scénario d'utilisation présenté dans la table 1. *Si vous constatez des divergences par rapport à votre réponse à la question précédente, alpaguez vos intervenants...*

TAB. 1 – Exemple de scénario nominal

Scénario "Poste de navigation"
Pour se rendre à un endroit donné, le marin va devoir déplacer son bateau. Cette tâche consiste en deux opérations : d'une part régler le cap et d'autre part régler la vitesse. Le marin règle le cap en l'augmentant ou en le diminuant et, de la même façon, il augmente ou diminue la vitesse. Sur ces embarcations, la vitesse maximale atteinte est de 25 noeuds.

Question 2 *Dans le texte du scénario :*

- *surlignez en jaune ce qui relève des concepts (données du domaine, propriétés des données).*
- *surlignez en vert la tâche abstraite, qui relève de l'intention du marin.*
- *surlignez en bleu les indicateurs de la(les) tâche(s) concrète(s) et des sous-tâches. Attention, les actions d'interaction sur le système, ou de dialoque ne font pas partie de la tâche (rayez ces parties dans le scénario).*

1.3 Arbre des tâches

Un arbre des tâches décrit les étapes de l'exécution du scénario, et ordonne la tâche en niveaux d'abstractions, du plus conceptuel au plus concret. Considérons par exemple l'arbre donné en Figure 1.

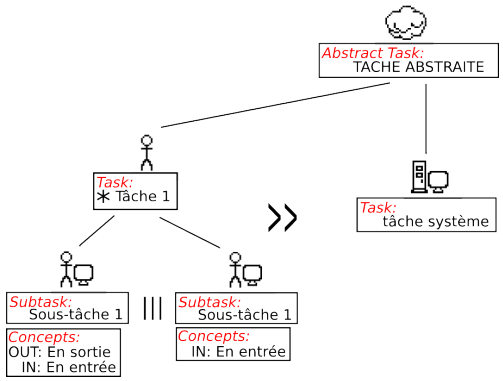


FIG. 1 – exemple d'arbre des tâches

Des opérateurs entre chaque branche d'un même niveau de l'arbre organisent l'ordonnancement et l'exécution des tâches.

- "T1 >> T2" : T1 est effectuée avant T2,
- "T1 ||| T2" : T1 et T2 peuvent être effectuées dansn'importe quel ordre,
- "*T1" : T1 est itérative, on peut l'effectuer plusieurs fois de suite,
- "[T1]" : T1 est optionelle.

D'autres opérateurs existent, mais ne seront pas utilisés pour la suite de la conception. Se référer pour être complet à [1].

Chaque tâche ou sous-tâche est exécutée par un utilisateur seul (*tâche utilisateur*), par le système seul (*tâche système*), ou par l'utilisateur et le système (*tâche interactive*). Dans l'exemple de la Figure 1, une tâche abstraite est réalisée par deux tâches concrètes, dont celle de l'utilisateur qui se décompose en deux tâches interactives. Ces deux sous-tâches interactives peuvent être effectuées dans n'importe quel ordre, mais elles doivent être réalisées toutes les deux pour valider la tâche utilisateur. Chaque validation de la tâche 1 déclenche l'activation d'une tâche système.

Pour chaque tâche, il est possible et même souhaitable d'associer les concepts nécessaires à la réalisation de la tâche. Dans l'exemple, les sous-tâches manipulent des concepts (probablement des données du système) en entrée ou en sortie.

TAB. 2 – Scénario nominal étiqueté

Scénario "Poste de navigation"
Pour se rendre à un endroit donné, le marin va devoir déplacer son **bateau**. Cette tâche consiste en deux opérations : d'une part régler **le cap** et d'autre part régler **la vitesse**. ~~Le marin règle le cap en l'augmentant ou en le diminuant et, de la même façon, il augmente ou diminue la vitesse~~. Sur ces embarcations, la **vitesse maximale** atteinte est de 25 noeuds.

Question 3 *En considérant le scénario étiqueté de la table 2, écrivez l'arbre des tâches correspondant. Décorez chaque tâche des concepts associés et n'hésitez pas à intervenir si vous aviez étiqueté le scénario de façon différente.*

2 Modèle-Vue-Composant

D'après Wikipedia [3], un "Modèle-Vue-Contrôleur est une architecture [...] qui organise l'interface Homme-Machine d'une application logicielle. Il divise l'IHM en un modèle (modèle de données), une vue (présentation, interface utilisateur) et un contrôleur (logique de contrôle, gestion des événements)". Nous allons implémenter le système de navigation selon cette architecture.

2.1 Description de l'architecture en Java

(D'après [2]). La séparation des données, de l'IHM et de la logique de contrôle impose une architecture en 3 couches :

- Le modèle : Il représente les données de l'application. Il définit aussi l'interaction avec la base de données et le traitement de ces données.
- La vue : Elle représente l'interface utilisateur, ce avec quoi il interagit. Elle n'effectue aucun traitement, elle se contente simplement d'afficher les données que lui fournit le modèle. Il peut tout à fait y avoir plusieurs vues qui présentent les données d'un même modèle.

– Le contrôleur : Il gère l'interface entre le modèle et le client. Il va interpréter la requête de ce dernier pour lui envoyer la vue correspondante. Il effectue la synchronisation entre le modèle et la(les) vue(s).

Ce modèle permet le changement d'une couche sans altérer les autres. Comme toutes les couches sont clairement séparées, on doit pouvoir en changer une pour, par exemple, remplacer Swing par SWT sans porter atteinte aux autres couches. On pourrait aussi donc changer le modèle sans toucher à la vue et au contrôleur. Cela rend les modifications plus simples.

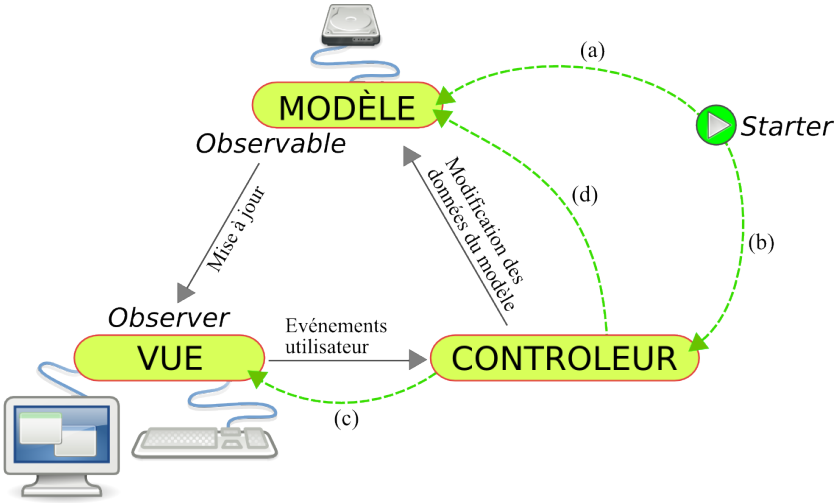
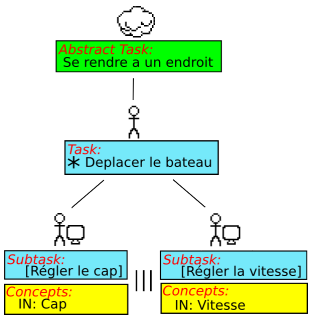


FIG. 2 – Architecture MVC implémentée en java

La figure 2 décrit le modèle tel que nous allons l'implémenter en Java. À l'exécution, les actions des utilisateurs sont renvoyées vers le contrôleur qui va modifier le modèle. Le modèle modifié va envoyer des événements sur le mode du pattern "*observer-observable*" aux vues enregistrées pour qu'elles se mettent à jour.

Au démarrage, un objet "*starter*" va instancier le Modèle (Fig. 2(a)), puis le Contrôleur (Fig. 2(b)). Le Contrôleur va instancier une ou plusieurs Vues (Fig. 2(c)) et la/les enregistrer dans la liste d'objets qui écoutent le modèle (Fig. 2(d)).

2.2 Retour sur l'arbre des tâches



La tâche de déplacement va s'organiser en deux sous-tâches toutes deux facultatives et non ordonnées l'une par rapport à l'autre. Dans un cas, le marin va commander au système de régler le cap du navire, dans l'autre cas, il réglera la vitesse. Ainsi, la tâche "déplacer le bateau" peut s'effectuer de 5 façons différentes :

– “Régler Cap” puis “Régler Vitesse”
– “Régler Vitesse” puis “Régler Cap”
– “Régler Cap”
– “Régler Vitesse”
– ne rien faire !

L’arbre des tâches doit ouvrir la voie pour choisir le style de dialogue et la façon de découper la présentation. Dans notre cas, s’agissant de la conception d’une interface graphique, il est important que la présentation reflète la décomposition de la tâche. Nous aurons deux parties distinctes à l’affichage, l’une pour le réglage du cap, l’autre pour le réglage de la vitesse. Comme ces sous-tâches ne sont pas ordonnées, nous savons que ces vues apparaîtront simultanément à l’écran. Enfin, le choix des interacteurs se porte sur des boutons “+” et “-” pour

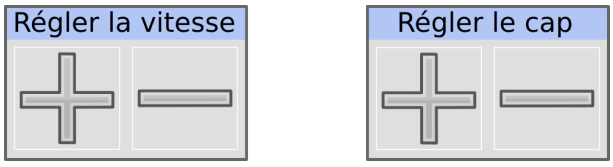


Fig. 3 – Proposition de maquette pour le système de navigation

régler les valeurs numériques – un peu comme pour le poste de navigation dont le marin à l’habitude. Avec ces quelques informations supplémentaires, il est possible de maquetter l’interface utilisateur de l’application (Fig. 3). Pour bien faire, il faudrait critiquer ces maquettes avec des utilisateurs et des ergonomes ; mais comme Concepteur 1 & 2 sont confiants, ils décident de proposer le produit fini aux marins et passent directement à l’implémentation.

3 Implementation dans Éclipse

Effectuez ces étapes pour importer le début de projet dans Éclipse :

1. Téléchargez le fichier “MVC-stage1.zip” sur Moodle et décompressez le dans votre répertoire personnel,
2. Dans Éclipse, faites “File”→“new”→“Project”, puis “Java project” et “Next”,
3. Donnez lui le nom de “MVC-stage1” et sélectionnez “Create project from existing source”.
4. Recherchez et sélectionnez le fichier décompressé “MVC-stage1”, puis validez l’importation par “Finish”

Exécutez le *starter*. Vous devez voir l’interface de gestion de réglage du cap s’afficher.

3.1 Creation du MVC de contrôle de la vitesse

Nous allons décomposer le système en deux MVC qui vont partager la brique du modèle. Les noms proposés pour les classes dans la figure 4 seront à respecter dans vos sources. Sur la figure, on décompte en tout 12 classes (avec le *starter*, non représenté). Six d’entre elles, qui correspondent à la partie Cap/Heading

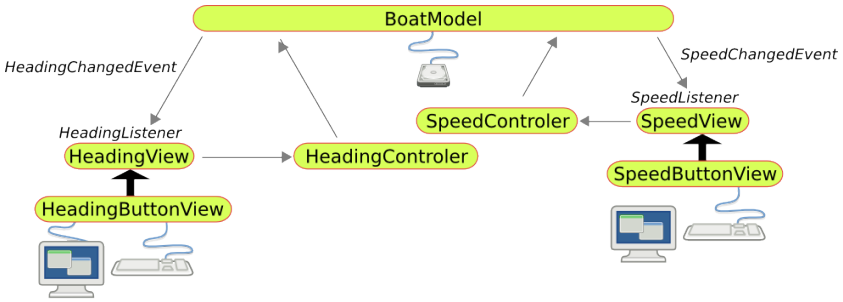


FIG. 4 – Organisation des classes du projet dans le MVC

du système de navigation, vous sont données dans les sources du projet. Celui-ci se découpe en 3 paquets (*controler, model et view*) ou sont rangées les classes du modèle MVC. Le dernier paquet contient Starter.java, le point d'entrée du programme.

Question 4 *En vous inspirant fortement du MVC de gestion du cap, créez et complétez les classes manquantes du MVC de gestion de la vitesse. Vous trouverez des commentaires dans les classes existantes qui vous décrirons les détails du fonctionnement. Procédez dans cet ordre :*

- *Créez et complétez l'interface "SpeedListener" d'après l'interface "headingListener",*
- *Créez et complétez la classe "SpeedChangedEvent" d'après la classe "HeadingChangedEvent",*
- *Créez et complétez la classe abstraite "SpeedView" d'après la classe "HeadingView",*
- *Créez et complétez la classe d'interface "SpeedButtonView" d'après la classe "HeadingButtonView",*
- *Créez et complétez la classe du contrôleur "SpeedControler" d'après la classe "HeadingControler",*
- *Complétez le corps de la classe "BoatModel" pour qu'elle s'intègre dans le MVC de la vitesse,*
- *Dé-commentez les commentaires dans la classe "Starter" et tentez d'exécuter le projet.*

Si vous n'arrivez pas à faire fonctionner votre code, qu'il y a des erreurs de compilation ou que vous ne comprenez pas certaines parties du code, appelez vos intervenants.

3.2 Désillusion et second prototype

Cela fait maintenant un mois que le prototype a été installé pour test sur les bateaux des deux marins qui avaient participé au premier entretien. Pour avoir un avis objectif sur la valeur du système, celui-ci à été mis en concurrence avec "l'ancien" système de navigation. Pour les deux concepteurs, c'est le moment du dé-briefing.

CONCEPTEUR 1: Bonjour à tous, Cela fait un mois que nous avons installé le prototype en plus de votre système de commande, nous voudrions savoir si vous êtes convaincus de son utilité

MARIN 1: Je ne l'utilise pas.

CONCEPTEUR 2: Comment ça ? Nous avons passé du temps à en discutter ensemble, et vous sembliez d'accord ...

MARIN 2: Moi non plus je ne l'utilise pas. C'est qu'il nous manque l'essentiel avec votre système ...

MARIN 1: ... oui, à aucun moment on ne sait à quelle vitesse ni à quel cap on navigue. pour cela on se réfère aux anciens instrument.

MARIN 2: En puis si c'est pour jongler de l'un à l'autre, autant laisser tomber le nouveau système.

CONCEPTEUR 1: Attendez, vous avez pourtant été clair en décrivant votre tâche. Il était juste question de régler le cap et la vitesse ; et c'est précisément ce que fait le système !

MARIN 2: C'est pourtant évident que nous devons contrôler la vitesse à un moment donné, non ?

CONCEPTEUR 2: Sans doute, mais vous savez, nous ne sommes pas marins..

CONCEPTEUR 1: Il va falloir que l'on passe à nouveau un moment pour mieux cerner la tâche.

Peut-être vous êtes vous fait la remarque en construisant l'interface ; c'est vrai qu'il manque quelque chose au prototype. Cet exemple simpliste doit vous faire prendre conscience que lorsque vous discutez avec des utilisateurs pour la conception d'un système, chaque partie reste dans son monde. En particulier, les concepts, les tâches, tout ce que l'on pense évident n'est en général pas énoncé. Il est de la responsabilité du concepteur de poser toute les questions pour éclaircir chacune des facettes de la tâche avec un maximum de détails.

Après une seconde analyse, le scénario a été reconstruit et l'arbre des tâches résultanat est présenté en figure 5. Il intègre des nouvelles sous-tâches, et le retour d'information est matérialisé sous la forme de deux tâches système qui succèdent au réglage des valeurs de cap et vitesse par l'utilisateur. Au niveau

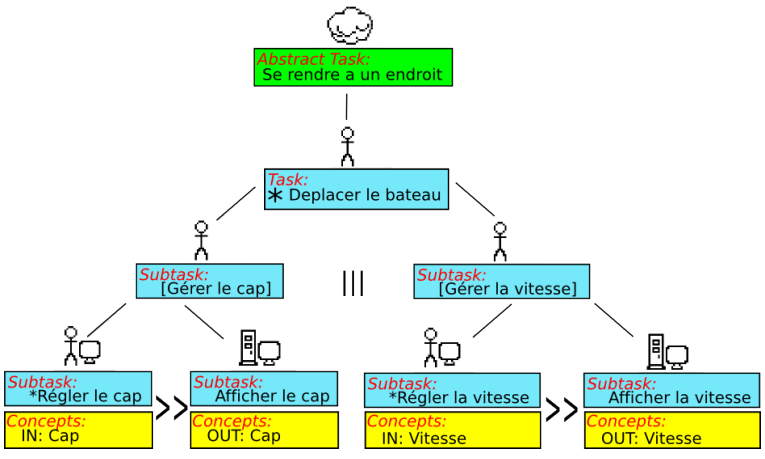


FIG. 5 – Arbre des tâches après une nouvelle analyse *(extrait)*

de la construction de l'interface, il est toujours possible d'identifier clairement deux sous-parties dans la tâche de déplacement de bateau. Le modèle n'est pas

modifié en profondeur et adapter un élément de présentation doit être suffisant pour intégrer le retour du système. Le moyen le plus simple est de modifier les vues du modèle pour qu'elles affichent la valeur du cap ou de la vitesse lorsque le modèle notifie un changement (Fig. 6).

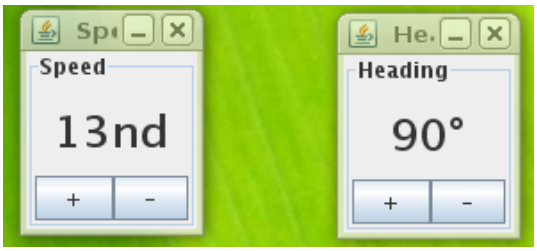


FIG. 6 – Seconde maquette du système de navigation

Question 5 *En vous inspirant de la maquette proposée en figure 6, vous ajouterez l'affichage des valeurs de cap aux fenêtres de réglage. Procédez comme suit (cas du cap)*[1] *:*

- *ajoutez un JLabel nommé "headingDisplay" en variable interne de la classe "HeadingButtonView",*
- *dans la méthode buildFrame, initialisez le "headingDisplay" en appelant le constructeur* `new JLabel(heading)`*,*
- *ajouter "headingDisplay" au "contentPane" (méthode* `add`*),*
- *dans la méthode HeadingChanged, mettre à jour le texte du Jlabel par un appel à la méthode* `setText` *de "headingDisplay"*

Procédez à l'identique pour l'affichage de la vitesse et testez. Au passage, vérifiez que le cap boucle entre 0° et 360°, et que la vitesse varie entre 0nd et 25nd. Si ce n'est pas le cas, corrigez le code dans BoatModel.

Ce prototype est à nouveau mis en test auprès des marins et cette fois-çi le dé-briefing semble plus positif :

CONCEPTEUR 1: Alors on me rapporte que vous arrivez à utiliser le prototype cette fois ?
MARIN 1: Oui, c'est beaucoup mieux ...
MARIN 2: Pour sûr, c'est simple à utiliser ...
CONCEPTEUR 2: Alors tout va bien ? Nous allons bientôt pouvoir vous remplacer ces archaïques systèmes de commandes.
MARIN 2: Mouais, faut voir.
MARIN 1: Y'aurait peut être encore un petit quelque chose, mais ça ...
CONCEPTEUR 2: ... mais ça ? ...
MARIN 2: ... oui ça ...
CONCEPTEUR 1: ... Ça, ce sera pour la prochaine fois !

[1] Pour obtenir le résultat maquetté, il faut ajouter un `Borderlayout` et un `TitledBorder` au `contentPane`, le code de la méthode `buildFrame` correspondante est donnée en annexe.

Références

[1] G. Mori, F. Paterno, and C. Santoro. CTTE : support for developing and analyzing task models for interactive system design. IEEE Transactions on Software Engineering, 28(8) :797–813, aug 2002.

[2] B. Wicht. Implémentation du pattern mvc. Technical report, April 2007.

[3] Wikipedia. http ://fr.wikipedia.org/wiki/mod Technical report, May 2008.

4 ANNEXE : Code de la méthode BuildFrame()

```
 private void buildFrame(){
        frame=new JFrame();
        frame.setResizable(false);
        contentPane=new JPanel();
        contentPane.setLayout(new BorderLayout());
        contentPane.setBorder(
                        BorderFactory.createTitledBorder("Heading"));

        plus=new JButton("+");
        plus.addActionListener(new PlusActionListener());
        minus=new JButton("-");
        minus.addActionListener(new MinusActionListener());

        headingDisplay=new JLabel("<html><H1>"+heading+
                        "&deg</H1></html>",JLabel.CENTER);

        contentPane.add(headingDisplay,BorderLayout.NORTH);
        contentPane.add(plus,BorderLayout.LINE_START);
        contentPane.add(minus,BorderLayout.LINE_END);

        frame.setContentPane(contentPane);
        frame.setTitle("Heading setup");
        frame.pack();
}
```