Projet Qualité Logicielle : Modèles des opérations

Équipe H4314 – 9 février 2014

Chef de projet
Jean-Marie COMETS

Membres de l'équipe
Franck MPEMBA BONI: responsable qualité
Pierre TURPIN
Samuel CARENSAC
Grégoire CATTAN
Van PHAN HAU
Iler VIRARAGAVANE

Table des matières

1 Système Balance

1.1 demandeMesure()

Description de l'opération $SystemeBalance :: demandeMesure()$:

Acteur du MdE : CapteurBalance
Cas d'utilisation : Obtenir Mesure
Messages :

* CapteurBalance::{mesure}

Pré-conditions :aucunes
Post-conditions :

```
if self.assign != none then
    self.mesure = 0.8 * self.mesure + 0.2 * mesure
    if self.mesure < self.seuilMin then
        self.SystemeWeb^alarm(self.quantite)
wait(1 minute)
self.demandeMesure()
```

1.2 scan()

Description de l'opération $SystemeBalance :: scan()$:

Acteur du MdE : Système externe
Cas d'utilisation : Installation balance
Messages :

* SystèmeExterne::{scan(clé)}

Pré-conditions :aucunes
Post-conditions :

```
if self.cleClient = cle then
    sender^confirmScan
    self.synchro = true
    self.SystemeWeb = sender
```

1.3 assignBal()

Description de l'opération $SystemeBalance :: assignBal()$:

Acteur du MdE : Système web
Cas d'utilisation : Assigner un produit à une balance
Messages :

* SystèmeWeb::{assignBal(Prod, Seuil, Quantité)}

Pré-conditions :

```
self.synchro = true
```

Post-conditions :

```
sender = self.SystemeWeb
if Seuil >= 0 and Quantite > 0 then
    self.assign = Prod
    self.seuilMin = Seuil
    self.quantite = Quantite
    self.mesure = 0
    sender^confirmAssign
else
    sender^errorAssign
```

1.4 configBal()

Description de l'opération $SystemeBalance :: configBal()$:

Acteur du MdE : Système web
Cas d'utilisation : Modifier un paramètre d'une balance
Messages :

* SystèmeWeb::{configBal(Seuil, Quantité)}

Pré-conditions :

```
self.synchro = true
self.produit != none
```

Post-conditions :

```
if Seuil >= 0 and Quantite > 0 then
    self.seuilMin = Seuil
    self.quantite = Quantite
    self.mesure = 0
    sender^confirmConfig
else
    sender^errorConfig
```

2 Receptacle

2.1 rentrerCode(n : number)

Description de l'opération $Receptacle :: rentrerCode(n : number)$:

Acteur du MdE : PaveNum
Cas d'utilisation : Réception d'une commande
Messages :

* porteLaterale::{debloquerPorte}
* panneauAffichage::{msgModifierCode, msgCodeIncorrect}
* diode::{diodeVerte, diodeRouge}

Pré-conditions :aucunes
Post-conditions :

```
if n = self.codeAdmin then
    self.mode = EnumMode::Admin
    self.porteLaterale^debloquerPorte()
    self.panneauAffichage^msgModifierCode(true)
    self.diode^diodeVerte()
else if n = self.codeUser then
    self.mode = EnumMode::User
    self.porteLaterale^debloquerPorte()
    self.panneauAffichage^msgModifierCode(true)
    self.diode^diodeVerte()
else
    self.panneauAffichage^msgCodeIncorrect(true)
    self.diode^diodeRouge()
```

2.2 modifierCode(n : number)

Description de l'opération $Receptacle :: modifierCode(n : number)$:

Acteur du MdE : PaveNum
Cas d'utilisation : Réception d'une commande
Messages :

* panneauAffichage::{msgModifiercode}

Pré-conditions :

```
not(self.porteLaterale.closedState)
```

Post-conditions :

```
if self.mode = EnumMode::Admin then
    self.codeAdmin = n
    self.panneauAffichage^msgModifierCode(false)
if self.mode = EnumMode::User then
    self.codeUser = n
    self.panneauAffichage^msgModifierCode(false)
```

2.3 verificationBadge(b : badge)

Description de l'opération $Receptacle :: verificationBadge(b : badge)$:

Acteur du MdE : capteurBadge
Cas d'utilisation : Réception d'une commande
Messages :

* porteLaterale::{debloquerPorte}
* panneauAffichage::{msgModifierCode, msgBadgeIncorrect}
* diode::{diodeVerte, diodeRouge}

Pré-conditions :aucunes
Post-conditions :

```
if b = self.idBadgeAdmin then
    self.mode = EnumMode::Admin
    self.porteLaterale^debloquerPorte()
    self.panneauAffichage^msgModifierCode(true)
    self.diode^diodeVerte()
else if b = self.idBadgeUser then
    self.mode = EnumMode::User
    self.porteLaterale^debloquerPorte()
    self.panneauAffichage^msgModifierCode(true)
    self.diode^diodeVerte()
else
    self.panneauAffichage^msgBadgeIncorrect(true)
    self.diode^diodeRouge()
```

2.4 verificationColis(idDestinataire : IdRFID)

Description de l'opération $Receptacle :: verificationColis(idDestinataire : IdRFID)$:

Acteur du MdE : systemeDrone
Cas d'utilisation : Livrer une commande
Messages :

* systemeDrone::{colisCorrect}

Pré-conditions :aucunes
Post-conditions :

```
if self.id = idDestinataire then
    self.systemeDrone^colisCorrect(true)
else
    self.systemeDrone^colisCorrect(false)
```

2.5 ouverturePanneau()

Description de l'opération $Receptacle :: ouverturePanneau()$:

Acteur du MdE : systemeDrone
Cas d'utilisation : Livrer une commande
Messages :

* panneauSuperieur::{ouvrirPanneau}
* systemeDrone::{sendConfirmation}

Pré-conditions :

```
not(self.panneauSuperieur.lockedState)
```

Post-conditions :

```
if self.panneauSuperieur^ouvrirPanneau() then
    self.panneauSuperieur.closedState = false
    self.systemeDrone^sendConfirmation(ConfirmMsg::PanneauOuvert)
```

2.6 fermeturePanneau()

Description de l'opération $Receptacle :: fermeturePanneau()$:

Acteur du MdE : systemeDrone
Cas d'utilisation : Livrer une commande
Messages :

* panneauSuperieur::{fermerPanneau}
* systemeDrone::{sendConfirmation}

Pré-conditions :

```
not(self.panneauSuperieur.lockedState)
```

Post-conditions :

```
if self.panneauSuperieur^fermerPanneau() then
    self.panneauSuperieur.closedState = true
    self.systemeDrone^sendConfirmation(ConfirmMsg::PanneauFerme)
```

2.7 timeoutVerification()

Description de l'opération $Receptacle :: timeoutVerification()$:

Acteur du MdE : systemeDrone
Cas d'utilisation : Livrer une commande
Messages :

* PanneauAffichage::{msgHorsService}

Pré-conditions :aucunes
Post-conditions :

```
self.panneauAffichage^msgHorsService(true)
```

2.8 dronePresent()

Description de l'opération $Receptacle :: dronePresent()$:

Acteur du MdE : supportDrone
Cas d'utilisation : Livrer une commande
Messages :

* supportDrone::{bloquerDrone}

Pré-conditions :

```
not(self.supportDrone.lockedState)
```

Post-conditions :

```
self.supportDrone^bloquerDrone(true)
self.supportDrone.lockedState = true
```

2.9 porteFerme()

Description de l'opération $Receptacle :: porteFerme()$:

Acteur du MdE : PorteLaterale
Cas d'utilisation : Livrer une commande
Messages :

* porteLaterale::{bloquerPorte}

Pré-conditions :aucunes
Post-conditions :

```
self.porteLaterale^bloquerporte()
self.porteLaterale.lockedState = true
```

2.10 panneauFerme()

Description de l'opération $Receptacle :: panneauFerme()$:

Acteur du MdE : PanneauSuperieur
Cas d'utilisation : Livrer une commande
Messages :

* panneauSuperieur::{bloquerPanneau}

Pré-conditions :aucunes
Post-conditions :

```
self.panneauSuperieur^bloquerPanneau()
self.panneauSuperieur.lockedState = true
```

2.11 colisPresent()

Description de l'opération $Receptacle :: colisPresent()$:

Acteur du MdE : capteurRFID
Cas d'utilisation : Livrer une commande
Messages :

* systemeClient::{signalerReception}

Pré-conditions :aucunes
Post-conditions :

```
self.systemeClient^signalerReception()
```

3 Système Ronde

3.1 cdeVisualiserDrone()

Description de l'opération $SystemeRonde :: cdeVisualiserDrone()$:

Acteur du MdE : Superviser Ronde
Cas d'utilisation :
Messages :

* ecranSupervision::{visualiserDrones}

Pré-conditions :aucunes
Post-conditions :

```
self.ecranSupervision^visualiserDrones()
```

3.2 cdeVisualiserTournées()

Description de l'opération $SystemeRonde :: cdeVisualiserTournes()$:

Acteur du MdE : Superviser Ronde
Cas d'utilisation :
Messages :

* ecranSupervision::{visualiserTournees}

Pré-conditions :aucunes
Post-conditions :

```
self.ecranSupervision^visualiserTournees()
```

3.3 envoyerCommande(cde : Commande)

Description de l'opération $SystemeRonde :: envoyerCommande(cde : Commande)$:

Acteur du MdE : Livrer une tournée
Cas d'utilisation :
Messages :

* BD::{ajouterCommande, calculerTournées}

Pré-conditions :aucunes
Post-conditions :

```
self.bd^ajouterCommande(cde)
self.bd^calculerTournees()
```

3.4 annulerCommade(lst : ListeCommande)

Description de l'opération $SystemeRonde :: annulerCommade(lst : ListeCommande)$:

Acteur du MdE : Livrer une tournée
Cas d'utilisation :
Messages :

* BD::{supprimerCommande, calculerTournees}

Pré-conditions :aucunes
Post-conditions :

```
self.bd^supprimerCommande(lst)
self.bd^calculerTournees()
```

3.5 notifierExpiration(t : Tournee)

Description de l'opération $SystemeRonde :: notifierExpiration(t : Tournee)$:

Acteur du MdE : Livrer une tournée
Cas d'utilisation :
Messages :

* drone::{envoyerDrone}
* BD::{supprimerCommande}
* boiteMail::{envoyerMessage}

Pré-conditions :aucunes
Post-conditions :

```
self.bd^supprimerCommande(sender.tournee.lstCommande)
sender.tournee.drone^envoyerDrone()
self.boiteMail^envoyerMessage(sender.tournee.drone.id, DEPART_DRONE)
sender.tournee.drone.oclInState(Livraison)
```

3.6 signalerPosition(p : Position)

Description de l'opération $SystemeRonde :: signalerPosition(p : Position)$:

Acteur du MdE : Livrer une tournée
Cas d'utilisation :
Messages :

* ecranSupervision::{visualiserDrone}
* BD::{majPositionDrone}

Pré-conditions :aucunes
Post-conditions :

```
self.bd^majPositionDrone(sender.drone.id, p)
self.ecranSupervision^visualiserDrone()
```

3.7 signalerLivraison(cde : Commande)

Description de l'opération $SystemeRonde :: signalerLivraison(cde : Commande)$:

Acteur du MdE : Livrer une tournée
Cas d'utilisation :
Messages :

* BD::{supprimerCommande}

Pré-conditions :aucunes
Post-conditions :

```
self.bd^supprimerCommande(cde)
```

3.8 signalerAnormalie(a : Anormalie)

Description de l'opération $SystemeRonde :: signalerAnormalie(a : Anormalie)$:

Acteur du MdE : Livrer une tournée
Cas d'utilisation :
Messages :

* BD::{majEtatDrone}
* boiteMail::{envoyerMessage}

Pré-conditions :aucunes
Post-conditions :

```
sender.drone.oclInState(Erreur)
self.bd.majEtatDrone(sender.drone.id, a)
self.boiteMail^envoyerMessage(sender.drone.id, a)
```

3.9 signalerRetour()

Description de l'opération $SystemeRonde :: signalerRetour()$:

Acteur du MdE : Livrer une tournée
Cas d'utilisation :
Messages :

* BD::{majEtatDrone}
* boiteMail::{envoyerMessage}

Pré-conditions :aucunes
Post-conditions :

```
self.bd.majEtatDrone(sender.drone.id, PRET)
self.boiteMail^envoyerMessage(sender.drone.id, RETOUR)
sender.drone.oclInState(Inactif)
```

3.10 estOperationnel()

Description de l'opération $SystemeRonde :: estOperationnel()$:

Acteur du MdE : Livrer une tournée
Cas d'utilisation :
Messages :

* BD::{majEtatDrone}
* DronePhysique::{envoyerDrone}

Pré-conditions :aucunes
Post-conditions :

```
self.bd.majEtatDrone(idDrone, BON_ETAT)
if sender.drone.Tournee.lstCommandes = none
    sender.drone.oclInState(Inactif)
    self.bd.majEtatDrone(idDrone, Pret)
else
    sender.drone.oclInState(Livraison)
    self.sender^envoyerDrone()
```

4 Drone

4.1 verificationColisActuel(idColis : IdRFID)

Description de l'opération $Drone :: verificationColisActuel(idColis : IdRFID)$:

Acteur du MdE : listeColis
Cas d'utilisation : Livrer une commande
Messages :

* systeReceptacle::{verificationColis}
* ListeTimers::{enclenchementTimer}

Pré-conditions :

```
self.receptacleConnecte != 0
```

Post-conditions :

```
self.systeReceptacle^verificationColis(self.listeColis[idColis].
    idDestinataire)
timeInfo.command = EnumCommand::verificationColis
self.ListeTimers^enclenchementTimer(timerInfo)
```

4.2 receptaclePlein()

Description de l'opération $Drone :: receptaclePlein()$:

Acteur du MdE : SystemeReceptacle
Cas d'utilisation : Livrer une commande
Messages :

* systèmeRonde::{SignalementErreur}

Pré-conditions :

```
self.receptacleConnecte != 0
```

Post-conditions :

```
self.systemeRonde^SignalementErreur("ReceptaclePlein")
self.etat = EnCirculation
```

4.3 colisCorrect(e : bool)

Description de l'opération $Drone :: colisCorrect(e : bool)$:

Acteur du MdE : SystemeReceptacle
Cas d'utilisation : Livrer une commande
Messages :

* systemeReceptacle::{ouverturePanneau}

Pré-conditions :

```
self.receptacleConnecte != 0
```

Post-conditions :

```
self.systemeReceptacle^ouverturePanneau()
```

4.4 signalReceptacle(idReceptacle : IdRFID)

Description de l'opération $Drone :: signalReceptacle(idReceptacle : IdRFID)$:

Acteur du MdE : SystemeReceptacle
Cas d'utilisation : Livrer une commande
Messages :aucuns
Pré-conditions :

```
self.receptacleConnecte = 0
```

Post-conditions :

```
self.receptacleConnecte = idReceptacle
self.etat = LivraisonEnCours
```

4.5 sendConfirmation(msg : ConfirmMsg)

Description de l'opération $Drone :: sendConfirmation(msg : ConfirmMsg)$:

Acteur du MdE : SystemeReceptacle
Cas d'utilisation : Livrer une commande
Messages :

* ListeTimers::{desactiverTimer}

Pré-conditions :

```
self.receptacleConnecte != 0
```

Post-conditions :

```
self.ListeTimers^desactiverTimer(msg)
```

4.6 expirationTimer(idTimer : IdTimer)

Description de l'opération $Drone :: expirationTimer(idTimer : IdTimer)$:

Acteur du MdE : ListeTimer
Cas d'utilisation : Livrer une commande
Messages :

* trappe::{activerSystUrgence}
* systemeRonde::{SygnalementErreur}

Pré-conditions :aucunes
Post-conditions :

```
if ListeTimers[idTimer].command = FermetureTrappe then
    self.trappe^activerSystUgence()
self.systemeRonde^SignalementErreur( ListeTimers[idTimer].command )
```

4.7 ColisPret()

Description de l'opération $Drone :: ColisPret()$:

Acteur du MdE : tapisRoulant
Cas d'utilisation : Livrer une commande
Messages :

* ListeColis::{getCurColis}

Pré-conditions :aucunes
Post-conditions :

```
self.ListeColis^getCurColis()
```

4.8 EnvoyerDrone(itineraire : ItineraireLivraison)

Description de l'opération $Drone :: EnvoyerDrone(itineraire : ItineraireLivraison)$:

Acteur du MdE : SystemeRonde
Cas d'utilisation : Livrer une commande
Messages :aucuns
Pré-conditions :aucunes
Post-conditions :

```
self.etat = EnCirculation
```

5 Système Web

5.1 voirCommande()

Description de l'opération $SystemeWeb :: voirCommande()$:

Acteur du MdE : BtnVoirCommande
Cas d'utilisation : Valider une commande
Messages :

* PanneauInformation::{afficherCommande, afficherFacture, produitNonDispo}

Pré-conditions :aucunes
Post-conditions :

```
if not(self.commande.estCommandeVide) then
    self.panneauInformation^afficherCommande(self.commande)
    and
    self.panneauInformation^afficherFacture(self.commande, self.commande.
        prixTotal)
    and
    self.commande.estCommandeValidee = false
    and
    if not(self.commande.listeArticle.article.estDisponible) then
        self.panneauInformation^produitNonDispo(self.commande.listeArticle.
            article, true)
    else
        self.panneauInformation^produitNonDispo(self.commande.listeArticle.
            article, false)
    endif
endif
```

5.2 alarm(art : Article, qte : Quantite)

Description de l'opération $SystemeWeb :: alarm(art : Article, qte : Quantite)$:

Acteur du MdE : SystèmeBalance
Cas d'utilisation : MAJ d'une commande
Messages :

* PanneauInformation::{produitNonDispo}
* ListeCommande::{majCommande}

Pré-conditions :aucunes
Post-conditions :

```
if art.estDisponible then
    if self.commande.estCommandeVide then
        self.commande.estCommandeVide = false
        and
        self.commande.listeArticle.article = art
        and
        self.commande.prixTotal = art.prixArticle * qte
    else
        if art = self.commande.listeArticle.article then
            self.commande.listeArticle.article.estDisponible = true
            and
            self.panneauInformation^produitNonDispo(self.commande.
                listeArticle.article, false)
            and
            self.commande.listeArticle.article.qteArticle = self.commande.
                listeArticle.article.qteArticle@pre + qte
        endif
        and
        self.commande.prixTotal = self.commande.prixTotal@pre + art.
            prixArticle * qte
        and
        self.commande.estCommandeVide = false
    endif
    and
    self.commande^majCommande()
else
    if self.commande.estCommandeVide then
        self.panneauInformation^produitNonDispo(art, true)
    else
        if art = self.article then
            self.commande.listeArticle.article.estDisponible = false
```

```
            and
            self.panneauInformation^produitNonDispo(self.commande.
                listeArticle.article, true)
        endif
    endif
    and
    self.commande^majCommande()
endif
```

5.3 produitNonStock(art : Article)

Description de l'opération $SystemeWeb :: produitNonStock(art : Article)$:

Acteur du MdE : InterfaceCatalogue
Cas d'utilisation : MAJ d'une commande
Messages :

* PanneauInformation::{produitNonDispo}

Pré-conditions :aucunes
Post-conditions :

```
if self.commande.listeArticle.article = art then
    self.commande.listeArticle.article.estDisponible = false
    and
    self.panneauInformation^produitNonDispo(self.commande.listeArticle.
        article, true)
endif
```

5.4 validerCommande()

Description de l'opération $SystemeWeb :: validerCommande()$:

Acteur du MdE : BtnValiderCommande
Cas d'utilisation : Valider une commande
Messages :

* PanneauInformation::{afficherCommande, afficherFacture, commandeValidée}
* ListeCommande::{majCommande}

Pré-conditions :aucunes
Post-conditions :

```
if not(self.commande.estCommandeVide) then
    self.panneauInformation^afficherCommande(self.commande)
    and
    self.panneauInformation^afficherFacture(self.commande, self.commande.
        prixTotal)
    and
    self.commande.estCommandeValidee = true
    and
    self.panneauInformation^commandeValidee(true)
    and
    self.commande^majCommande()
else
    self.panneauInformation^commandeValidee(false)
endif
```

5.5 changerQuantité(art : Article, qte : Quantité)

Description de l'opération $SystemeWeb :: changerQuantit(art : Article, qte : Quantit)$:

Acteur du MdE : ListeCommande
Cas d'utilisation : Valider une commande
Messages :

* PanneauInformation::{afficherCommande, commandeValidée}

* ListeCommande::{majCommande}

Pré-conditions :aucunes
Post-conditions :

```
if not(self.commande.estCommandeVide) then
    self.panneauInformation^afficherCommande(self.commande)
    and
    self.panneauInformation^commandeValidee(false)
    and
    if art = self.commande.listeArticle.article then
        self.commande.prixTotal = self.commande.prixTotal@pre + (qte - self
            .commande.listeArticle.article.qteArticle) * art.prixArticle
        and
        self.commande.listeArticle.article.qteArticle = qte
    endif
    and
    self.commande^majCommande()
endif
```

5.6 estArticleCoché(art : Article)

Description de l'opération $SystemeWeb :: estArticleCoch(art : Article)$:

Acteur du MdE : NavigateurWeb
Cas d'utilisation : Valider une commande
Messages :

* PanneauInformation::{afficherCommande, commandeValidée}
* ListeCommande::{majCommande}

Pré-conditions :aucunes
Post-conditions :

```
if not(self.commande.estCommandeVide) then
    self.panneauInformation^afficherCommande(self.commande)
    and
    self.panneauInformation^commandeValidee(false)
    and
    if art = self.commande.listeArticle.article then
        if self.commande.listeArticle.article.estCoche then
            self.commande.listeArticle.article.estCoche = false
            and
            self.commande.prixTotal = self.commande.prixTotal@pre - self.
                commande.listeArticle.article.qteArticle * art.prixArticle
        else
            self.commande.listeArticle.article.estCoche = true
            and
            self.commande.prixTotal = self.commande.prixTotal@pre + self.
                commande.listeArticle.article.qteArticle * art.prixArticle
        endif
        and
        self.commande^majCommande()
    endif
endif
```

5.7 configBal(bal : Balance, seuil : Seuil, qte : Quantite)

Description de l'opération $SystemeWeb :: configBal(bal : Balance, seuil : Seuil, qte : Quantite)$:

Acteur du MdE : BtnConfigBalance
Cas d'utilisation : Configurer les balances
Messages :

* PanneauInformation::{confirmConfigBal, erreurConfigBal}

Pré-conditions :aucunes
Post-conditions :

```
if (bal = self.balance and seuil <= 0 and qte <= 0) then
    self.panneau.information^erreurConfigBal()
else
    self.panneauInformation^confirmConfigBal()
endif
```

5.8 assignBal(bal : Balance, art : Article, seuil : Seuil, qte : Quantité)

Description de l'opération $SystemeWeb :: assignBal(bal : Balance, art : Article, seuil : Seuil, qte : Quantit)$:

Acteur du MdE : BtnAssignBalance
Cas d'utilisation : Configurer les balances
Messages :

* PanneauInformation::{afficherBalance, confirmAssignBal, erreurAssignBal}

Pré-conditions :aucunes
Post-conditions :

```
self.panneauInformation^afficherBalance(bal)
and
if (bal = none and art = self.commande.listeArticle.article and seuil <= 0
   and qte <= 0) then
    self.panneau.information^erreurAssignBal()
else
    self.panneauInformation^confirmAssignBal()
endif
```