Étude de cas « *GCC*: Gestion des Communications d'une conférence »

Denis Conan

CSC4102

Télécom SudParis Janvier 2025

1 Introduction à l'étude de cas

Pour partager et évaluer les résultats de recherche, la communauté scientifique présente et publie des communications dans des conférences et leurs actes (version électronique ou papier). Ces communications (en anglais, « conference papers ») sont soumises pour être évaluées par les pairs, c'est-à-dire d'autres chercheurs et chercheuses du même domaine de recherche. L'objectif du système que nous nous proposons de construire est de gérer le processus de soumission et d'évaluation des communications d'une conférence[1]. Le logiciel se limite à la gestion d'une seule conférence et est ainsi nommé GCC[2].

La solution GCC permet :

- au président ou à la présidente du comité de programme, qui est constitué des évaluateurs et évaluatrices, de gérer la conférence : fixation de l'agenda, ajout des évaluateurs et évaluatrices, répartition des évaluations, prise de décision (communication acceptée ou non), annonce des décisions aux auteurs et autrices, etc. ;
- aux auteurs et aux autrices, de soumettre des communications à la conférence et d'être informés de la décision (« communication acceptée » ou « communication non acceptée »[3]) ;
- aux évaluateurs et aux évaluatrices, d'évaluer les communications en rendant des avis accompagnés de rapports.

La section qui suit contient le cahier des charges de l'application à réaliser.

2 Cahier des charges

C'est l'administrateur du système qui ajoute le président ou la présidente du comité de programme. Ensuite, c'est le président ou la présidente qui constitue le comité de programme en s'y incluant. Par contre, lorsqu'un groupe d'auteurs et d'autrices souhaite soumettre une communication, chacun d'eux peut s'ajouter au système et ajouter les coauteurs. Les informations communes à tous les utilisateurs sont un identifiant, un nom, un prénom, et une institution (modélisée en une chaîne de caractères).

Lors de la création du système, les noms et l'agenda de la conférence sont fixés. L'agenda de la conférence est constitué de la date limite de soumission des communications (d_s), la date d'annonce des décisions suite aux évaluations (d_a), la date limite de préparation de la version finale des communications acceptées (d_p), et la période de déroulement de la conférence (entre d_d et d_f) avec $d_s < d_a < d_p < d_d \leq d_f$. On simplifie l'étude de cas en favorisant les comparaisons strictes et en limitant les égalités entre les dates. Avant d_s (incluse), c'est la phase de soumission ; dans l'intervalle de temps $]d_s, d_a[$, c'est la phase d'évaluation ; et dans l'intervalle de temps $[d_a, d_p]$, c'est la phase de révision. Important : les auteurs et autrices peuvent abandonner leur communication à tout moment ; l'abandon est une opération définitive ; et une communication abandonnée n'est pas retirée du système.

Pendant la phase de soumission, la communication est dite « en cours de soumission » car les auteurs et autrices peuvent modifier les informations sur la communication : le titre, la liste des coauteurs, le résumé, et le contenu (modélisé en une chaîne de caractères). Un auteur ou une autrice peut soumettre plusieurs communications. En outre, par déontologie, dans certaines conférences comme ICT4S[4], les membres du comité de programme ne peuvent pas soumettre de communication. Nous prendrons cette hypothèse simplificatrice pour GCC.

Pendant la phase d'évaluation, c'est le président ou la présidente du comité de programme qui répartit les communications aux évaluateurs et évaluatrices. Dans les systèmes existants, il est possible de déclarer des

1. Version très simplifiée inspirée de « HotCRP » (`https://hotcrp.com/`) et de « EasyChair » (`https://easychair.org/conference`).
2. Ce logiciel est un objet malicieux pour « Gestion des Communications d'une Conférence ». Toute dissemblance avec un logiciel réel, passé ou présent, est intentionnelle et volontaire.
3. Contrairement aux processus d'évaluation des journaux, le processus d'évaluation des communications d'une conférence est contraint dans le temps car la conférence est un événement programmé : il n'est pas possible de passer par des phases de révisions majeure ou mineure, avec ré-évaluation, etc.
4. ICT for Sustainability : `https://conf.researchr.org/series/ict4s`.

conflits d'intérêt : par exemple, je collabore dans mes recherches avec telle autrice et je déclare que je ne peux déontologiquement pas évaluer telle communication dont elle est coautrice. Cette première version de GCC ne gère pas de telles contraintes lors de la répartition des évaluations, qui est effectuée par le président ou la présidente. Les évaluations peuvent être ajoutées ou modifiées jusqu'à la date d'annonce des décisions d_a (non incluse) : par exemple, une évaluation supplémentaire peut être demandée pour confirmer ou infirmer une décision. Une évaluation inclut un avis (parmi « acceptation forte », « acceptation faible », « neutre », « non-acceptation faible », « non-acceptation forte ») ainsi qu'un rapport (modélisé en une chaîne de caractères).

À la fin de la phase d'évaluation, c'est le comité de programme qui décide, lors d'une réunion plénière, de la liste des communications acceptées, les autres étant non acceptées. En ce qui concerne GCC, c'est le président ou la présidente du comité de programme qui enregistre dans le système la décision, et on ne notifie pas les auteurs et autrices tant qu'une décision n'est pas rendue pour une des communications de la conférence. Dans certains cas particuliers, une décision peut être prise sans qu'il y ait d'évaluation sur la communication : par exemple, le sujet de la communication est en dehors des thèmes de la conférence.

Pendant la phase de révision, les auteurs et les autrices des communications acceptées peuvent modifier le titre, le résumé ainsi que le contenu de leurs communications, mais pas la liste des coauteurs. Pour simplifier l'étude de cas, les communications finalisées ne sont pas ré-évaluées.

Pour que l'effort de développement reste raisonnable, on n'essaiera pas dans le temps imparti de réaliser les fonctionnalités suivantes :

- **modification ou retrait de la présidence, d'un évaluateur, d'une évaluation, d'une décision, etc. ;**
- **abandon d'une communication.**

Dans la dernière partie du module, nous ajoutons un système de notification :

- au président ou à la présidente du comité de programme, (1) de la soumission d'une nouvelle communication, (2) du dépôt d'une nouvelle évaluation,
- aux évaluateurs et évaluatrices, d'une nouvelle évaluation à réaliser,
- aux auteurs et autrices, de la décision concernant leurs communications lorsque le président ou la présidente du comité de programme diffuse les décisions.

Important : ne cherchez pas à réaliser cette dernière partie avant qu'elle ne soit présentée et détaillée dans les énoncés des TP !

2.1 Complétude du logiciel

Comme indiqué dans les diapositives présentant le module (diapositive intitulée « Modalités d'évaluation »), la note du projet comprend 1 point sur la complétude du logiciel livré à la fin du module. Nous précisons donc ici ce que nous entendons par « logiciel complet » :

- les fonctionnalités suivantes sont programmées, et les tests de validations sont programmés et « passent » :
  - le processus complet de la soumission d'une communication à la notification de la décision, hors la fonctionnalité d'abandon,
  - le mécanisme de notification avec toutes les notifications demandées ;
- le code des classes, hors les classes de test, est complètement documenté ;
- l'intégration continue est mise en place avec la compilation, la vérification du style et de la qualité du code ainsi que l'exécution des tests. Par ailleurs, GitLab doit indiquer que « tout passe » (au vert).

A Scénario exemple d'utilisation du logiciel complet

Voici un scénario proposé pour la compréhension de l'étude de cas. Ce n'est en aucun cas un test de validation. Par conséquent, vous n'avez aucune obligation à le programmer. Dans ce scénario, la marque « ✓ » indique que l'opération est acceptée et effectuée, la marque « ✗ » indique que l'opération est refusée (non effectuée). **Attention !** Si vous n'avez pas un logiciel complet, adaptez le scénario à votre développement.

Création de la conférence

(1) date du test = aujourd'hui

(2) ✓ créer la conférence « *International Conference on Rhinolophus* » avec
date limite de soumission = aujourd'hui + 30,
date d'annonce des décisions = aujourd'hui + 60,
date de la version finale = aujourd'hui + 90,
date de début de la conférence = aujourd'hui + 120,
date de fin de la conférence = aujourd'hui + 123

(3) ✓ ajouter la présidente : identificateur = ourdis, nom = Ourdis-Cotdoussan, prénom = Émeline, du Conservatoire d'espaces naturelles

(4) ✗ ajouter le président : identificateur = autre, nom = Autre, prénom = Président, de l'institut Y
- *Il y a déjà une présidente*

Soumissions de communications

(5) ✓ ajouter un auteur : identificateur = arthur, nom = Arthur, prénom = Pascal, de l'institut Chauve-qui-peut

(6) ✓ soumettre une communication : auteur = arthur, titre = « Oreillard gris, Grand Murin, et Pipistrelle », ...

(7) ✓ ajouter un auteur : identificateur = auperman, nom = Auperman, prénom = Éric, du Muséum d'histoire naturelle

(8) ✓ ajouter un auteur à une communication : auteur = auperman, titre = Oreillard gris, Grand Murin, et Pipistrelle

(9) ajouter 25 jours à la date du test

(10) ✓ soumettre une communication : auteur = auperman, titre = Moyens-fers-à-cheval contre Petits-fers-à-cheval, ...

(11) ajouter 5 jours à la date du test

(12) ✓ ajouter un auteur : identificateur = courtois, nom = Courtois, prénom = Paul, de l'institut Centre des intrus

(13) ✓ soumettre une communication : auteur = courtois, titre = Le vrai coût énergétique du numérique, ...

(14) ✓ ajouter un auteur : identificateur = robiller, nom = Robiller, prénom = Sven, de University of California Davis

(15) ajouter 1 jour à la date du test

(16) ✗ soumettre une communication : auteur = robiller, titre = Après l'heure..., ...
- *la date limite de soumission est passée*

Constitution du comité de programme, évaluations, et décisions

(17) ✓ décider à propos d'une communication : présidente = ourdis, titre = Le vrai coût énergétique du numérique, décision = « non acceptée »

(18) ✗ notifier des décisions : présidente = ourdis
- *la date d'annonce n'est pas passée*

(19) ✓ ajouter une évaluatrice : présidente = ourdis, identificateur = svetlik, nom = Svetlik, prénom = Sophie, de la Maison de la Nature Auvergnate

(20) ✓ ajouter une évaluatrice : présidente = ourdis, identificateur = vignon, nom = Vignon, prénom = Élisabeth, du Pipistrellus Abramus Studies Center

(21) ✓ ajouter une évaluatrice à une communication : présidente = ourdis, évaluatrice = ourdis, titre = Oreillard gris, Grand Murin, et Pipistrelle

(22) ✓ ajouter une évaluatrice à une communication : présidente = ourdis, évaluatrice = svetlik, titre = Oreillard gris, Grand Murin, et Pipistrelle

(23) ✓ ajouter une évaluatrice à une communication : présidente = ourdis, évaluatrice = vignon, titre = Oreillard gris, Grand Murin, et Pipistrelle

(24) ✓ ajouter une évaluatrice à une communication : présidente = ourdis, évaluatrice = vignon, titre = Moyens-fers-à-cheval contre Petits-fers-à-cheval

(25) ✗ ajouter une évaluatrice à une communication : présidente = courtois, évaluatrice = svetlik, titre = Moyens-fers-à-cheval contre Petits-fers-à-cheval
- *courtois n'est pas présidente*

(26) ✓ ajouter une évaluation à une communication : évaluatrice = ourdis, titre = Oreillard gris, Grand Murin, et Pipistrelle, avis = « neutre », ...

(27) ✓ ajouter une évaluation à une communication : évaluatrice = svetlik, titre = Oreillard gris, Grand Murin, et Pipistrelle, avis = « acceptation forte », ...

(28) ✓ ajouter une évaluation à une communication : évaluatrice = vignon, titre = Moyens-fers-à-cheval contre Petits-fers-à-cheval, avis = « acceptation faible », ...

(29) ✗ ajouter une évaluation à une communication : évaluatrice = svetlik, titre = Moyens-fers-à-cheval contre Petits-fers-à-cheval, avis = « acceptation forte », ...
- *ce n'est pas une évaluatrice de cette communication*

(30) ✓ décider à propos d'une communication : présidente = ourdis, titre = Oreillard gris, Grand Murin, et Pipistrelle, décision = « acceptée »

(31) ✓ décider à propos d'une communication : présidente = ourdis, titre = Moyens-fers-à-cheval contre Petits-fers-à-cheval, décision = «acceptée »

(32) ajouter 30 jours à la date du test

(33) ✗ ajouter une évaluation à une communication : évaluatrice = ourdis, titre = Oreillard gris, Grand Murin, et Pipistrelle, avis = « acceptation forte », ...
- *la date limite d'évaluation = la date d'annonce est passée*

(34) ✗ décider à propos d'une communication : présidente = ourdis,

titre = Oreillard gris, Grand Murin, et Pipistrelle, décision = « acceptée »
- *la date limite de décision = la date d'annonce est passée*

Notifier les décisions
(35) ✓ notifier des décisions : présidente = ourdis

B Manipulation des dates avec la bibliothèque csc4102-util

Nous proposons une bibliothèque pour aider et accélérer la mise en œuvre de votre logiciel. Les quelques explications qui suivent sont à compléter avec la documentation JavaDoc des classes de la bibliothèque[5] ainsi qu'avec les exemples d'utilisation proposés dans le projet GitLabEnse csc4102-exemples.

B.1 Classe eu.telecomsudparis.csc4102.util.Datutil

Deux manières de dater sont proposées avec les classes java.time.LocalDate et java.time.Instant. Une date est un jour dans le temps avec le mois et l'année. Un instant est un point du temps à la précision de la nanoseconde. C'est *a priori* la classe LocalDate qui est la plus intéressante pour cette étude de cas. Voici donc quelques méthodes de la classe Datutil :

```
static LocalDate dateDuTest() // initialisé à LocalDate.now()
static LocalDate aujourdhui() // initialisé à dateDuTest()
static resetDateDuTest()      // à utiliser au début des méthodes @BeforeEach
static boolean   dateEstAujourdhui(LocalDate date)
static boolean   memeJour(LocalDate premiereDate, LocalDate secondeDate)
static boolean   dateEstAvantAujourdhui(LocalDate date)
static boolean   dateEstAvant(LocalDate premiereDate, LocalDate secondeDate)
static boolean   dateEstAvantOuAujourdhui(LocalDate date)
static boolean   dateEstApresAujourdhui(LocalDate date)
static boolean   dateEstApresOuAujourdhui(LocalDate date)
static void      ajouterJoursALaDateDuTest(final int nbjours)
static void      retirerJoursALaDateDuTest(final int nbjours)
static LocalDate ajouterJoursADate(LocalDate date, int nbJours)
static LocalDate retirerJoursADate(LocalDate date, int nbJours)
static String    dateToString(LocalDate date)

static Instant instantDuTest() // initialisé à Instant.now()
static Instant maintenant()    // initialisé à instantDuTest()
static resetInstantDuTest()    // à utiliser au début des méthodes @BeforeEach
static boolean memeInstant(Instant premierInstant, Instant secondInstant)
static boolean instantEstAvant(Instant premierInstant, Instant secondInstant)
static void    ajouterJoursALInstantDuTest(final int nbjours)
static void    ajouterJoursALInstantDuTest(int nbjours)
static void    ajouterSecondesALInstantDuTest(final long nbseconds)
static void    retirerSecondesALInstantDuTest(final int nbseconds)
static Instant ajouterJoursAInstant(Instant instant, int nbJours)
static Instant retirerJoursAInstant(Instant instant, int nbJours)
static Instant ajouterAInstant(Instant instant, ChronoUnit unite, int quantite)
static Instant retirerAInstant(Instant instant, ChronoUnit unite, int quantite)
static String instantToString(Instant instant)
```

On note donc plus particulièrement que les méthodes `aujourdhui`, `retirerJoursADate` et `ajouterJoursADate` peuvent servir à fixer l'agenda de la conférence, et une fois que les dates de l'agenda de la conférence sont fixées, que les méthodes `ajouterJoursALaDateDuTest` et `retirerJoursALaDateDuTest` peuvent servir à construire des scénarios de test avec des dates du test modifiées.

N.B. : comme indiqué, au début d'un scénario de test, il est préférable de remettre la date du test au jour d'exécution du test. Autrement dit, la méthode de test JUnit annotée `@BeforeEach` doit *a priori* commencer par l'instruction « `Datutil.resetDateDuTest();` ».

5. http://www-inf.telecom-sudparis.eu/COURS/CSC4102/maven-repository/site/apidocs/index.html