Compte rendu réunion R1.01

14 novembre 2022

Auteur : Rémi C.  
Participants : Anne, Rémi S., Isabelle, Insaf, Lionnel, Franck, François, Thierry, Rémi C. et Marguerite,   
Excusée : Bénédicte,   
Invités absents : Nicolas, Dominique

# Préambule

## Définitions (personnelles)

**Algorithmique** :

Méthodes et/ou activités permettant de construire la logique de résolution d’un *problème* (traitement des données) à l’aide d’un code informatique.

Dans les méthodes formelles classiques, on retrouve : l’axiomatique, la sémantique opérationnelle ou dénotationnelle.

**Codage :**

Transformer un algorithme en fichier interprétable par une machine : écrire des lignes de codes.

**Développement :**

A partir de l’énoncé d’un problème, **concevoir l’algorithme** de résolution, les tests permettant de tester l’algorithme et le code associé, **écrire le code**, et analyser les résultats des tests.

## Choix des IUTs info

Premièrement, la compétence visée étant le développement (algo, codage, test), deuxièmement, les langages de développement ayant un pouvoir d’expression suffisant pour se passer des langages de spécifications des algorithmes, **les ressources ne font plus de distinctions entre** : algorithmique, codage ou test.

# Constat

Les étudiants ont du mal :

1. Avec l’algorithmique, certains pensent maîtriser suffisamment le codage (python) pour se passer de l’algorithmique ;
2. Avec la logique, ils ne maîtrisent pas les notions du carré logique d’Aristote (contraire, contradictoire, subalterne, subcontraire).

Ces difficultés se traduisent par un manque de travail et de motivation en première année, par des lacunes en 2ème année (incapacité d’écrire une insertion dans une liste triée).

1. Certains items n’ont pas été abordés avant la SAé

# Questions

## Intérêt de la piscine ?

L’intérêt de la piscine est questionné, un simple test de logique permettrait-il de faire le même travail ?

*Rémi C.*: en dehors de l’aspect groupes de niveau, la piscine permet de faire découvrir une grande partie des compétences qui seront vues sur les 3 années. La dimension « guide de voyage : ce que vous allez découvrir lors de passage à l’IUT, me parait essentiel, typique de l’enseignement supérieur ».

## Juste un problème logistique ?

*Marguerite* : Ces difficultés sont-elles juste un problème de volumes horaires ?

* *Franck*: On ne règle pas un problème de méthode, d’approche pédagogique par plus d’heures.

Ou un problème de pédagogie ?

Lionnel précise qu’il ne faut pas séparer théorie et pratique

# Propositions

## Suppression du langage support à l’algorithmique, LAP

*Rémi C.* : L’intérêt du LAP disparait du fait que les langages de programmations ont aujourd’hui une capacité d’expression équivalente (typage, paramètres, fonction, structure de contrôles). On n’utilise plus de l’assembleur, ni des langages avec des goto, jump, ni des numéros de lignes !

De plus, le LAP ne permet pas de faire de l’algorithmique, c’est juste un langage de programmation avec sa propre syntaxe et sémantique (celle du pascal).

## Choix d’un langage pour débuter, afin de mettre les étudiants sur le même niveau.

*Insaf* : Pascal ?

* *Rémi C.* : Pascal c’est du LAP avec un compilateur

*Isabelle* : C ? beaucoup d’entreprises utilisent encore C, Fortran, voire Cobol ?

* *Rémi C.* : Je ne crois pas que revenir à un langage impératif non-objet soit une solution, quid du passage à la programmation objet, à la programmation par prototype (javascript) ou fonctionnel !

*Rémi C.* : un langage permettant d’écrire de l’axiomatique, donc de faire de l’algorithmique comme COQ

* *Réponse globale* : C’est quoi COQ ? 😊
* *Rémi C.* : c’est un langage qui permet d’écrire des propriétés axiomatiques, des preuves, en plus du code.Par exemple : on peut définir des propriétés, on considère un type X, muni de 1 opérateurs @, on peut décrire les propriétés d’existence d’un élément neutre vis-à-vis de @ :

|  |
| --- |
| **Lemma**  y\_neutre : ((**exists** y : X), (**forall** x: X, (x = x @ y) ). |

Et/ou une notion de modularité par rapport à @ :

|  |
| --- |
| **Lemma**  X\_modulaire : (**forall** x : X, (**exists** p: X, (x = p @ p))). |

*Rémi C.* : un langage fonctionnel ?

* *Isabelle* : Trop compliqué pour les étudiants !

## Changement d’approche

*Rémi C.* : commencer par de la modélisation UML light ?

# Conclusion

* Constat partagé, besoin d’adaptation pour ne pas dire changement
* Pas de décision, besoin de faire une autre réunion !

Préparation réunion 2

Proposée par Rémi C., destinée à rassembler des contributions et à être discuté sans modération.

# Plan d’actions

## Définir ce qu’on attend des étudiants à l’issue de R1.01.

*Rémi C.* : les compétences du BUT : **concevoir et expliquer un algorithme**.

## Rassembler les propositions

|  |  |
| --- | --- |
| **Proposition** | **Commentaires & remarques** |
| **Changer de langage pour débuter**   * C * Java * Javascript * Un langage fonctionnel : SCALA, Clojure, Elixir | Mettre les étudiants sur un plan d’égalité.  *Rémi C.* :   * **C** : d’un point de vue algorithmique, trop de forme syntaxique peuvent faire des choses non conformes avec une axiomatique claire * **Java/ Javascript** : on risque de retomber sur les problèmes de python. * **SCALA** : la solution proposée par l’EPFL pour notre problème, le langage est conçu comme un LAP propre du 21ème siècle permettant de faire objet & fonctionnel, * **Clojure** : son lien avec le développement web (capacité de transpilé en js) est très intéressant ! * **Elixir** : le plus exotique, donc mon préféré ! |
| **Commencer par de la modélisation/algorithmique** | Obliger les étudiants à réfléchir : qu’elles sont les données permettant de modéliser mon problème, quelles relations entre les données, comment traiter les données : flux, séquences, concurrence. Le passage au code peut être délicat pour certains. |
|  |  |
|  |  |

# Ethique et Gouvernance

**Règle 1 :** nous nous inscrivons dans le Programme National.

**Règle 2**: les choix possibles qui ont des impacts sur d’autres ressources, doivent être concertés avec les parties prenantes.

**Règle 3** : Code de l’enseignement supérieur : tous les enseignant.s.es et enseignan.ts.es-chercheur.s.es ont la capacité d’intervenir dans tous les enseignements généraux de leur champ disciplinaire tel que défini par les sections du Conseil national des universités.

Deux conséquences directes :

* Aucun choix ne peut être vu comme la volonté d’exclusion de personnes ne maitrisant pas tel ou tel langage, approche, etc. Exemple : Nous sommes tous compétent.s.es pour faire du Clojure.
* Aucun choix ne peut avoir comme fondement : « je ne maîtrise pas ce langage, approche, etc. ». Exemple : Je suis contre Clojure, car je n’ai jamais développé en Clojure, n’est pas recevable.

Contrepartie :

* Le choix peut demander du temps de préparation, et nécessiter une planification dans sa mise en œuvre.

**Règle 4 :** les décisions doivent être transparentes.

**Règle 5**: le processus de prise de décision est à définir mais doit intégrer les règles précédentes.