Université Montpellier-II
UFR des Sciences - Département Informatique
Master Informatique - UE HMIN305

Méta-Programmation et Réflexivité.

PROJETS

Projet en deux étapes.

1 Etape 1

L'étape 1 sera réalisée en TP à partir de la semaine du vendredi 22 novembre puis par travail individuel. Vous pouvez bien sûr commencer à y travailler dés maintenant.

A rendre : avant les congés de Noël, m'envoyer par mail un Zip contenant votre étude.

Au choix 1 ou 2 :

1. Réaliser l'exercice de construction d'un noyau objet réflexif, selon le programme guidé présenté au chapitre 2 du livre "A simple reflective Object Kernel" (`http://books.pharo.org/booklet-ReflectiveCore`), également accessible ici : `notes-etudes/reflective-objectKernel.pdf`.
   A rendre : votre code commenté.
2. Etude et extension de COMPO : un langage réflexif de programmation et de description d'architecture à composants décrit en particulier dans cet article : Petr Spacek, Christophe Dony, and Chouki Tibermacine. A Component-based meta-level architecture and prototypical implementation of a Reflective Component-based Programming and Modeling language. In Procs. of ACM CBSE'14 - The 17th Int. ACM SIGSOFT Conf. on Component Based Software Engineering, pages 13-23. July 2014 : ici
   Tous documents : ici.
   Différentes possibilités, étude ou implantation (dont portage du noyau bootstrappé de la version 6 à la version 7 de Pharo). Possibilité de poursuivre en stage recherche.

2 Etape 2

Etape 2 : à choisir une projet dans une des deux catégories "Langages" ou "Problématiques".

Faire un Compte-Rendu de lecture ou d'étude (de 2 à 5 (ou plus) pages) reflétant : votre compréhension de l'article lu, et/ou toute idée que la lecture vous a inspiré, et/ou toute implantation dans le langage de votre choix que la lecture vous a donné envie de réaliser.

Modalités : Rendre le possiblement la date d'examen. Choisir un projet individuel. J'inscrirai les étudiants au fur et à mesure, premier arrivé, premier servi.

3 Orientation "Langages"

Reprendre les exercices des TDs/TPs dans le langage (capacitaire) de votre choix (Javascript, Python, Objective-C, Php, Ruby, etc). Voir également `http://newspeaklanguage.org/`.

A rendre un texte de 3 à 5 pages (ou plus), incluant des exemples décrits et associés à un code commenté. Indiquez si nécessaire le lien pour obtenir le langage ou l'environnement utilisé pour les tests.

4 Orientation “problématiques”

Il s’agit ici :

— soit de lire un article ou un document et d’en produire un résumé intéressant de deux pages minimum décrivant une idée clé que vous avez comprise, illustrée par un exemple,

— soit d’essayer un système ou un langage et d’en produire un résumé descriptif avec un exemple de code.

Voici ci-dessous des sujets ouverts à l’étude.

4.1 Tester Reflexion à “load-time” avec Javassist

Shigeru Chiba, ”Load-time Structural Reflection in Java,” ECOOP 2000 Object-Oriented Programming, LNCS 1850, Springer Verlag, page 313-336, 2000.

https://en.wikipedia.org/wiki/Javassist

http://www.javassist.org/

article à lire : notes-etudes/Javassist.pdf

4.2 Les intergiciels réflexifs

Donmaine : Adaptive and Reflexive Middleware. http://cedric.cnam.fr/arm.

A lire (en partie) : Thomas Ledoux, “Reconfiguration dynamique d’architectures logicielles : des métaclasses aux “nuages verts””, https://tel.archives-ouvertes.fr/tel-01864344.

4.3 Variations sur l’adaptation dynamique pour les non spécialistes des systèmes réflexifs

Software Engineering of Self-Adaptive Systems : An Organised Tour and Future Challenges, notes-etudes/2017-SE-SAS.pdf.

Reflecting on Self-Adaptive Software Systems, notes-etudes/Reflecting-Self-Adaptive-Software-Systems.pdf.

Modeling and Analyzing MAPE-K Feedback Loops for Self-adaptation, notes-etudes/Make-K-Loop.pdf.

4.4 Miroirs versus Méta-Objets

Mirrors : Design Principles for Meta-level Facilities of Object-Oriented Programming Languages. Gilad Bracha, David Ungar, notes-etudes/mirrors.pdf

Voir aussi : Towards Structural Decomposition of Reflection with Mirrors. Nick Papoulias, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, Luc Fabresse.

4.5 Approfondir la question de la composition et de la compatibilité des méta-classes

Une proposition pour mieux utiliser les métaclasses explicites. Noury M. N. Bouraqadi-Saâdani, Thomas Ledoux, Fred Rivard : Safe Metaclass Programming. OOPSLA 1998 : notes-etudes/Bour98a-SafeMetaclassProg.pdf

4.6 Adaptation non anticipée de comportement

Comment réaliser des logiciels que l’on peut faire évoluer à l’exécution, notamment Dynamic Software Update ...

these-S-Costiou.pdf

4.7 Interpréteur réflexif

Réaliser un interpréteur capable d'interpréter entièrement son propre code : `notes-etudes/ReflectiveInterpreter.pdf`

4.8 Explication de la démonstration du théorême de Gödel

Le théorème de Gödel montre que tout système formel contenant a minima l'arithmétique de Peano autorise l'expression d'énoncés indémontrables. Sa démonstration par l'absurde est réalisée en encodant dans un tel système formel une expression contenant une contradiction. Cet encodage nécessite un plongement du méta-niveau dans le niveau de base et cela relève donc conceptuellement de ce cours.

Le site `http://www.felderbooks.com/papers/godel.html` propose une explication synthétique et claire de cette démonstration, reprenant l'explication fournie par Douglas Hofstadter dans son livre "Gödel, Escher, Bach : les brins d'une guirlande éternelle

4.9 Etc, etc

Si un autre sujet lié à la méta-programmation et réflexivité vous intéresse, faites moi une suggestion.