Premier cours

LISP = langage Traitement de listes ((langage de base IA))

ADA = multi processeur

SIMULA = Simulation ((simulation = notion de temps, on prévoit que la variable vaudras … à tel moment))

70 smalltak = introduit POO

Programmation orientée donnée => introduit les variables dynamiques qui sont modifié par des procédure(démon)

Formalisme DEVS => Générer des algorithmes de simulation automatiquement

Concevoir un nouveau langage

Un langage de programmation est un langage moins riche et moins ambigu qu’un langage humain

1ère approche par compilateur

Compilateur = traduit le langage humain en langage machine

Étapes de compilation

Analyse du langage humain

Envoie d'éventuel message erreur

Si pas d’erreur généré le code machine équivalent

2eme approche créer un langage à partir d’un langage existant (langage interprété)

IA => au début des année 50 l’idée était de créer une machine aussi intelligente que l’homme

Cela étant impossible l’idée a ensuite été de segmenter les domaines de connaissance (d’essayer de faire des programmes spécialisés pour un problème)

En informatique les représentations sont des approximations imparfaites de la réalité.

=>car dans nos représentations on tient compte de certaines choses, et d'autres non.

La représentation des connaissances est inspirée d’autre sciences tels que : math, biologie, économie, statistique, psychologie

Statistique introduit la notion d’incertitude entre 40 et 60

Principe d’encapsulation = regroupement de données et procédure (classes)

En informatique on distingue deux entité le code et les données

L’encapsulation permet qu’une procédure ne peut accéder qu’au donnée d’une encapsulation.

À l'extérieur d’une encapsulation on en voit qu’un nom unique (la classe) est des points d'entrée(méthodes). A l'intérieur on voit des données locales structurées.

Arbre d’héritage => dans POO tout objet du système que l’on crée est rattaché à un objet supérieur (arbre d’objets) (classe prédéfinie = objets)

POO héritage => un seul parent contrairement à l’IA (par soucis de simplification et de test)

Frames

Les frames sont une représentation des faits essentiels d’une entité.

Les parties des frames sont représentés comme slot ((attributs POO))

Les frames sont liés à d’autres frames

Facette => valeur par défaut, exception,

Facette comme procédure (démon) se déclenche dans certains événement (par ex-changement de valeur du slot)

RPO (IA)

Frames

Les frames sont une représentation des faits essentiels d’une entité.

Les parties des frames sont représentés comme slot ((attributs POO))

Les frames sont liés à d’autres frames

Facette => valeur par défaut, exception,

Facette procédurale (démon) se déclenche dans certains événement ((par ex-changement de valeur du slot))

La facette permet d’être plus fin dans la représentation de l’objets

La puissance = 2 ((valeur par défaut))

La valeur réel = 8 ((valeur réel obtenus par rapport aux données dans du contexte)

La classe définit deux aspects

Structurelle ((les variables peuvent avoir une ou plusieurs valeur)

Inférentielle ((modification des valeurs par les démons))

IA = multi spécialisation deux classe parent

Cela peut engendrer des conflits si deux parents ont le même nom d’attributs mais pas les mêmes contraintes

Objets composite un objet un la composition d’autres élément une voiture = volant roue carrosserie

IA = multi spécialisation deux classe parent

Cela peut engendrer des conflits si deux parents ont le même nom d’attributs mais pas les mêmes contraintes

Il existe plusieurs stratégies pour résoudre ces conflits.

Les principales stratégies en i en z et en N

i => en considérant que les valeurs exactes

z => par hiérarchie (on va du bas vers le haut) ((une valeur par défaut plus proche de l’objets (plus bas dans l’arbre) est privilégia a une valeur plus exacte mais plus éloigné))

En n => les valeurs exactes sont privilégiées

Objets composite un objet un la composition d’autres élément une voiture = volant roue carrosserie

Composite = héritage horizontale

Réflexe => d’abords on supprime ou ajoute la donnée puis les démons se déclenche

\*frames

\*slots \*face \* value = this.slot, this.face etc…