Interface Homme-Machine

Philippe Gaussier
Alexandre Pitti

Plan du cours

1 Introduction à l'IHM, historique et ergonomie

- psychologie, handicap et théorie, méthodes de conception et d'évaluation

2 Éléments d'une IHM

- Interfaces et Infrastructures, prototypage

3 Extraction et Traitement de l'information

- GUI Visualisation de l'information, Méthodes statistiques

4 & 5 Techniques d'interactions avancées

- Réalité Augmentée, Interface Tangible, projection 3D,
- Analyse du mouvement

Interfaces des IHMs

Multiplicité des interfaces:

- écran/souris clavier
- parole (synthèse reconnaissance)
- tablette/écran tactiles
- image (naturelle, capteurs IR) : WII, Kinect, Sony…, les systèmes de motion capture (lourdeur, équipement de la personne, salle dédiée, …)
- sons
- accélérations, orientation (WII, retour d'effort (phantom…)
- RFID
- EEG, BMI

Infrastructure pour les IHMs : IVY bus intelligent pour les IHMs

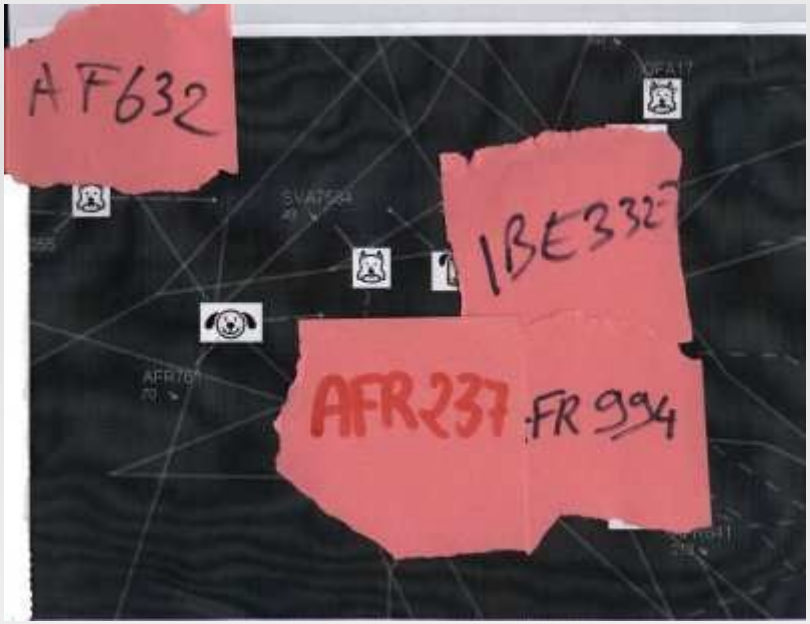
Pourquoi Ivy?

- Fédérer des composants IHM et ATC
- Mode de développement
  - Prototypage itératif
  - Travail en équipe
  - Langages différents
  - Étudier de nouveaux moyens d'interaction sur des plateformes différentes

Prototypage d'IHM

- Fédérer des composants IHM
- Mode de développement
  - Prototypage itératif
  - Travail en équipe
  - Langages différents
  - Étudier de nouveaux moyens d’interaction sur des plateformes différentes

Des méthodes de prototypage d'IHM



- Centré utilisateur, itératif
- étape cognitive
- Basse fidélité / Haute fidélité
- Itérations rapides
- Laisser la conception **émerger**

Prototypage → développement

Difficultés

- Pas d’outil miracle « taille unique »
- Réutiliser le savoir faire sans y être confiné
- Niveau d’abstraction des échanges entre concepteurs
- Pas forcément de plate-forme commune
- Les outils sont souvent liés à leur plate-forme
- La « synergie » entre les composants ne se fait pas.

→ Briser la barrière du prototypage

Une solution : le bus logiciel

- Principe : communication interprocessus
  - ne pas rester au niveau de la *socket*
  - métaphore du bus : agents, messages, abonnements
- Exemples
  Koalatalk, tooltalk, OAA, AppleEvents, SOAP, Corba

- Inconvénients
  - centralisation
  - cout d'apprentissage élevé
  - plate-formes spécifiques
  - incompatibilité des modèles d'architecture et d’exécution

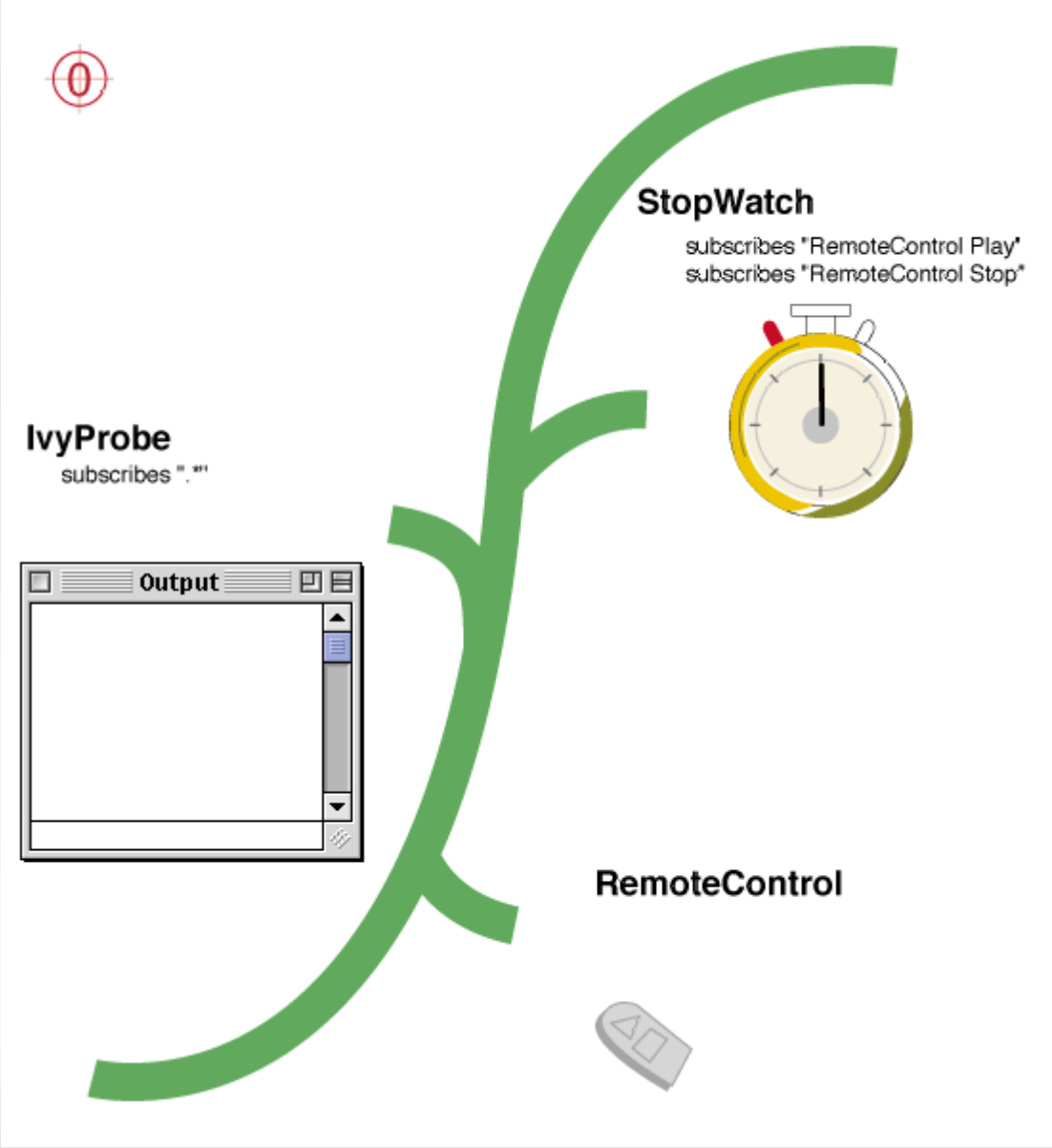
le bus logiciel Ivy

- 5 ans d'utilisation
- Environnements de démonstration
- Environnement de bureau
- vers une communauté autour du logiciel libre

Centre d'Etudes de la Navigation Aérienne

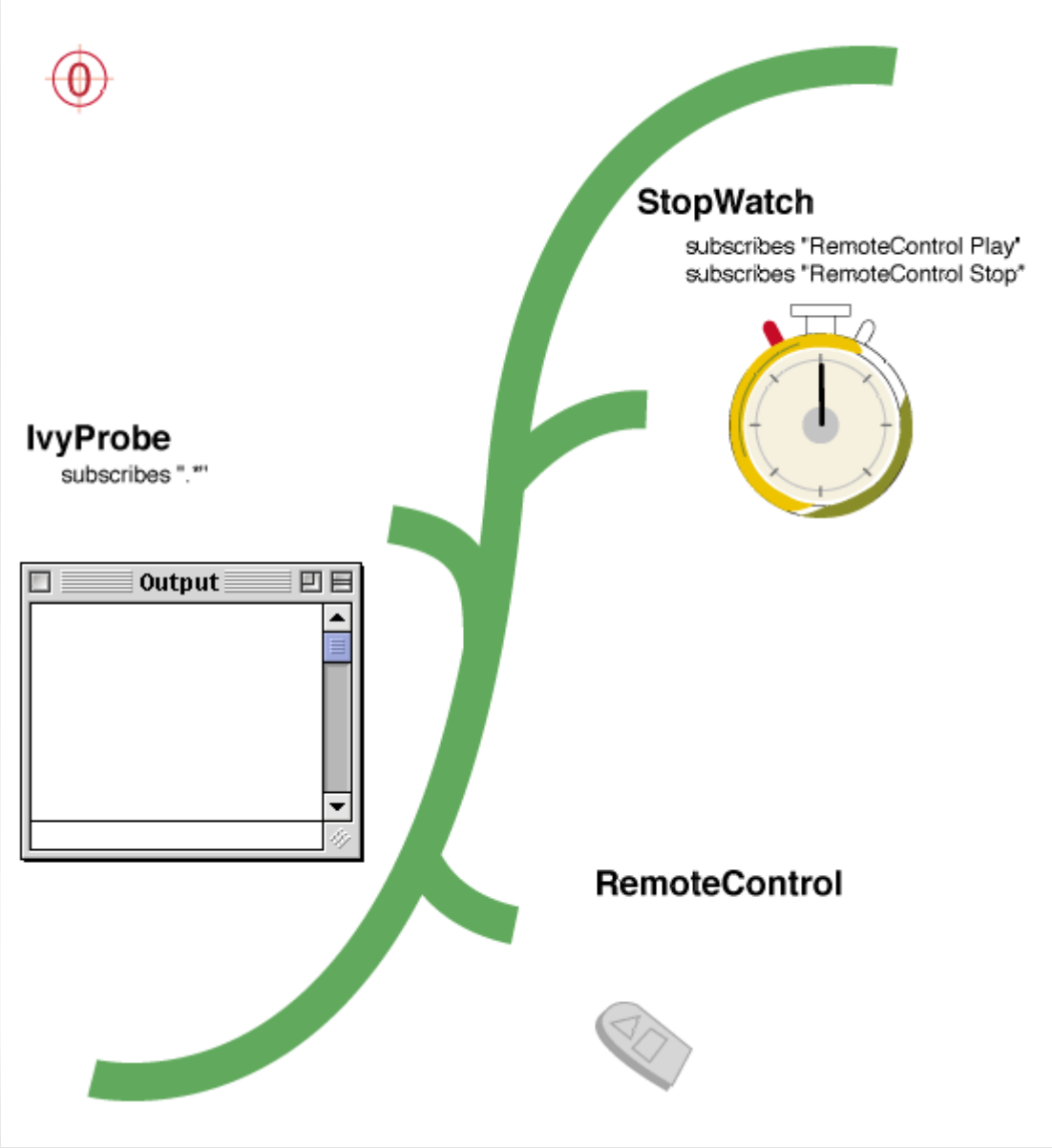
http://www2.tls.cena.fr/products/ivy/

Principe de fonctionnement



- Pas de serveur centralisé
- Communication par messages Textuels (conventions)
- Abonnement sur expression régulières
- Exécution de comportement à la réception des messages

Principe de fonctionnement



This Ivy bus is joined by three agents:

IvyProbe, commonly spawned in order to monitor the messages during development.

StopWatch, a toy watch subscribed to 'RemoteControl (Start|Stop)'.

RemoteControl is an agent connected to an infrared device.

The timeline is as following:

RemoteControl sends "RemoteControl Start" on the bus, IvyProbe will receive "RemoteControl Start" and display this in the window, and StopWatch will receive "Start" and start running

RemoteControl sends "RemoteControl Stop" on the bus, IvyProbe will receive "RemoteControl Stop" and display this in the window, StopWatch will receive "Stop" and compute the elapsed time (in this example, 27 seconds),

StopWatch will send "StopWatch 0:27", RemoteControl will receive this message and display it.

Exemple d'utilisation

- Agents :
  - télécommande
  - applications existantes (powerpoint, firefox, acrobat reader)
- Messages : commandes simples
- Comment rajouter un style d'interaction : la reconnaissance de geste
- Transparence du réseau, l'ordinateur disparait

Style de programmation

```
Ivy monbus= new Ivy(...) ;

monbus.start(...) ;

monbus.bindMsg("(.*)",callback) ;
```

- Modèle d’exécution événementiel :
  - simple à comprendre
  - inclus dans les boucles d’événements des boites à outils utilisées
  - langage de commande, paramètre sans structure

Quelques agents

- En entrée
  - reconnaissance de la parole, du geste, de la position
  - infra rouge, boite à boutons, tablette midi, stylo Anoto
  - système de fichier, IRC, …
- En sortie
  - synthèse vocale, rejeu audio
  - événements X, « on screen display »
- Applications « métier »
  - images radar, rejeu radar, flight simulator, simulation de traitement de plan de vol, trafic réel, météo
  - gestion d’expérimentation

L'existant

Unix/Linux, Win32, WinCE, MacOS, Java

C, C++, Java, Javascript, TCL, Perl, Python, ADA, CAML, COM

Outils existants de développement :

- ivyprobe, ivymon, enregistrement et rejeu de scénarios
- traduction de messages
- réutilisation facile de tout le logiciel déjà développé

Retour d’expérience

Quelles mesures ?

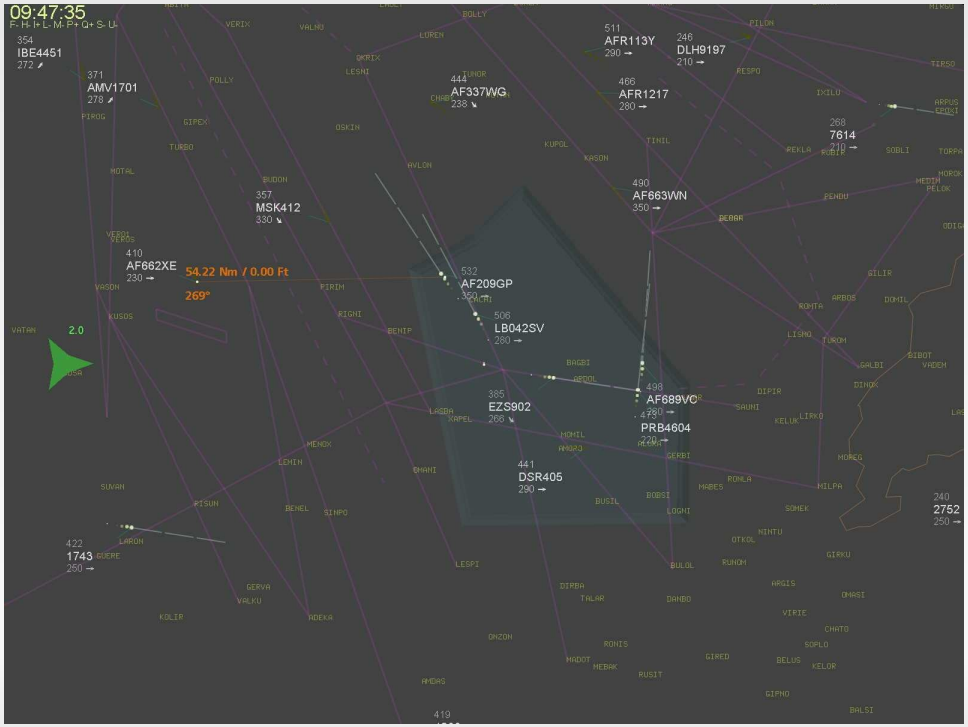
- Nombre de concepteurs, nombre d'agents, d'applications, nombre et types de messages
- Temps d'apprentissage
- Temps de développement

Non mesurable

- Satisfaction du programmeur
- Potentiel synergique

Analyse d'accident

Développement itératif
centré utilisateur



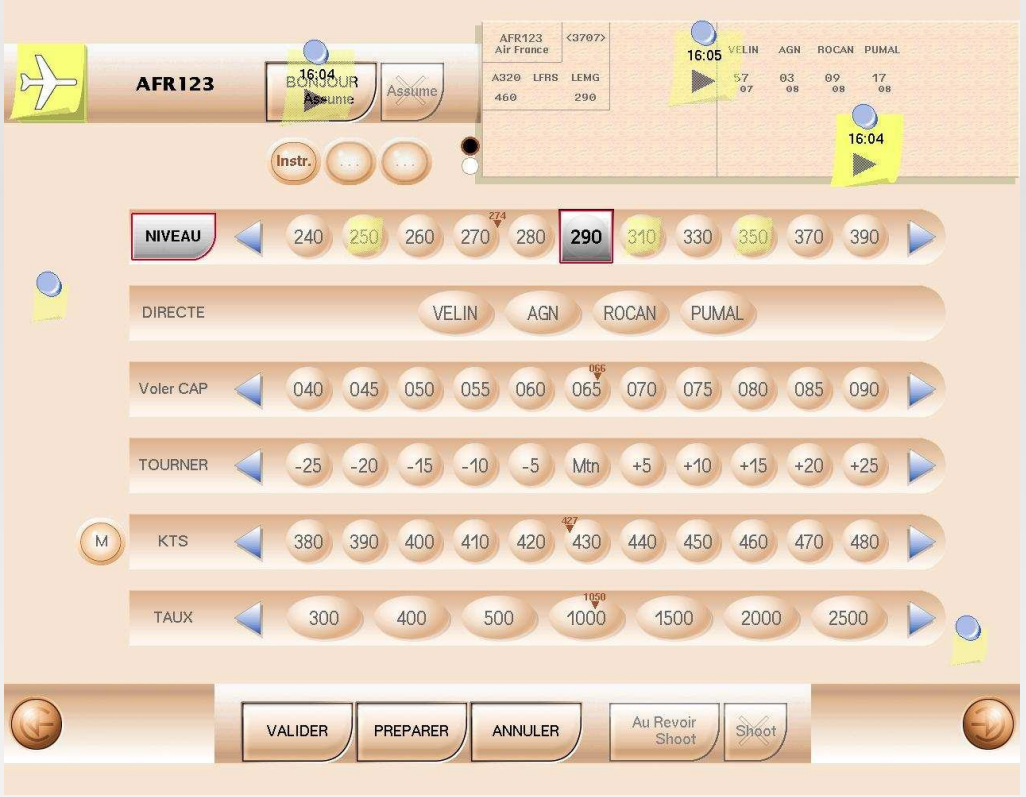
Exemples : analyse, RF & Post-it vocaux

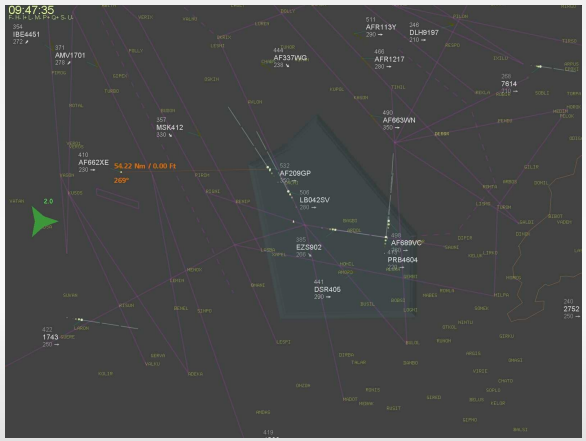
Conception : 1 mois
Développement : 3 semaines

Apprentissage ivy : ½ journée

Prototypage : 1 semaine

Développement itératif
centré utilisateur





Limites

– Vocabulaire des messages à inventer

– Nommer les agents

– Modèle événements

– Performances ?

– Expressions régulières

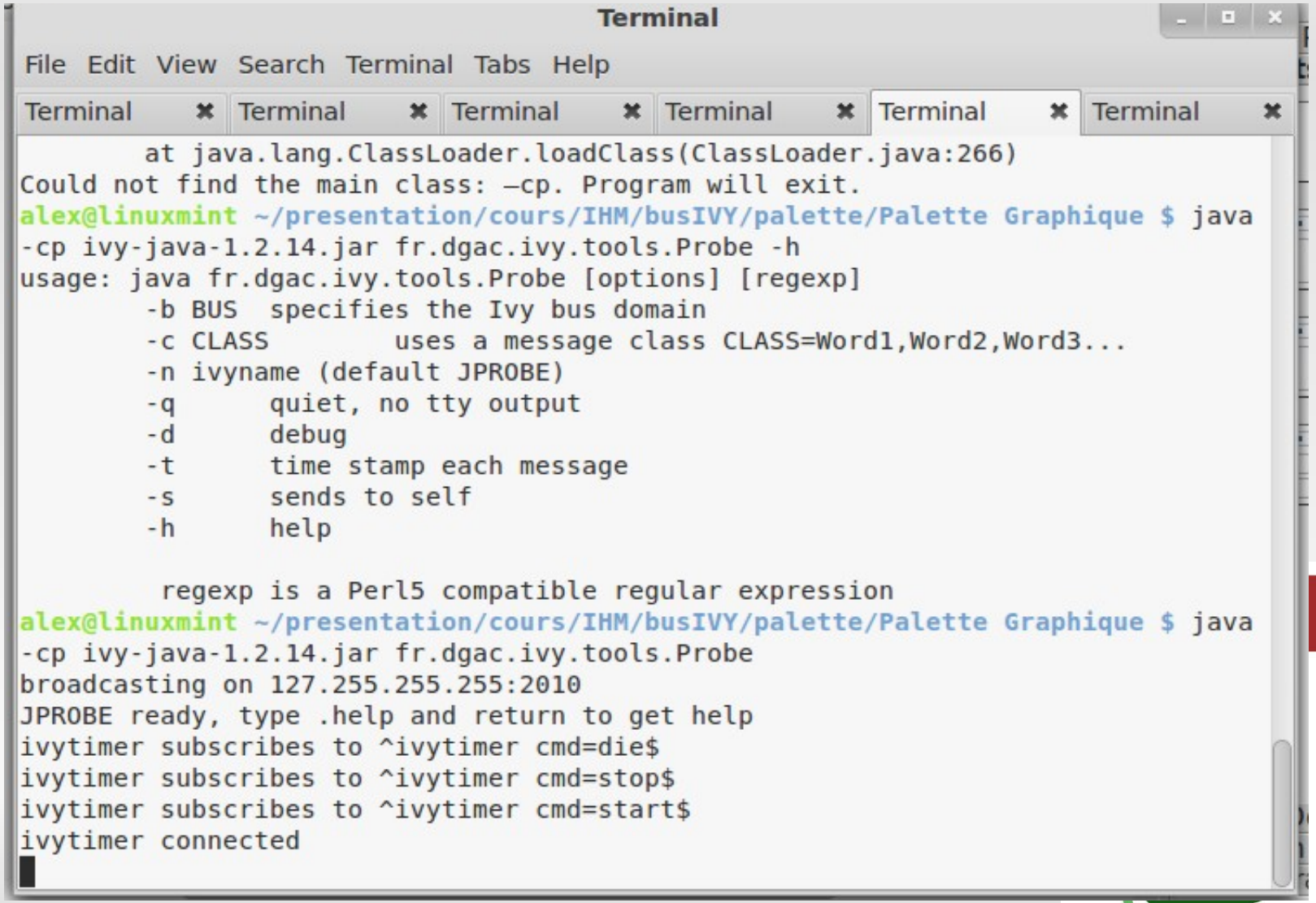
prototypage

Conclusions

- Fonctionnalités
  - Mécanisme de communication entre objets interactifs
  - Connaissance faible ou inexistante des autres objets
- Support au travail en équipe
  - Simplicité d'apprentissage
  - Ouvert aux nouveaux composants
  - Respecte le savoir faire existant
- Support à l'innovation
  - Interaction repartie
  - Interaction multimodale
  - Contexte

ivyprobe

java -cp ivy-java-1.2.14.jar fr.dgac.ivy.tools.Probe



programmation en C

- Du point de vue d'un programmeur, Ivy est un canal de flux d'information. Les fonctions principales sont:

  - Connexion au bus.
  - Envoie d'un message.
  - S'abonner a un type de message avec une fonction de retour (callback).
  - Fermeture de la connexion
  - Lancement de la boucle principale

1 Création d'un bus

- Initialisation d'Ivy avec la bibliotheque Ivy C se fait en deux etapes. D'abord, initialiser les bibliotheques en appelant la fonction IvyInit.

- IvyInit ("IvyTranslater", "Hello world", 0, 0, 0, 0);

  Quand le programme va démarrer,

  l'agent « IvyTranslater » va apparaître sur le bus et va envoyer le message « Hello world».

- L'agent n'est pas encore connecté au bus

2 Connexion au bus

- La connexion au bus se fait sur le port du reseau local (adresse IP). Cette adresse est composée en 2 parties: IP_address:port.
- Si cette variable n'est pas définie, la valeur par défaut "127.255.255.255:2010" est utilisée, port '2010' par défaut.
-
- La fonction IvyStart connecte votre application au bus.
- IvyStart ("127.255.255.255:2010");
- Ivy est lancée sur le réseau à l'adresse 127.255.255.255 sur le port 2010.

3 Envoie de messages

- Émettre un message sur le bus Ivy bus est comme afficher un message à l'écran. Néanmoins, n'oubliez pas que votre message ne sera pas émis si Ivy n'est pas initialisé correctement ou s'il n'y a pas de boucle principale qui tourne.
- Pour émettre un message, utilisez IvySendMsg, qui fonctionne comme printf:
  - strcpy(arg, "Robert") ;
  - IvySendMsg ("Hello %s", arg);
- Le message "Hello Robert" est envoyé sur le bus.

4 Réception de messages

- Souscrire à des messages consiste à coupler une fonction de retour (callback) a un message. Les expressions régulières sont :
- Quand un message dans le bus coïncide avec l'expression régulière de la fonction callback.
- Utilisez la fonction IvyBindMsg pour coupler une fonction callback à une expression régulière, et la fonction IvyUnbindMsg pour supprimer ce couplage.
  - IvyBindMsg (HelloCallback, 0, "^Hello (.*)");
- Tous les messages de forme « Hello qqch » appelle la fonction HelloCallback.

5 Fermeture de la session

- La fonction IvyStop va déconnecter l'application de la boucle principale.
  - IvyStop();
- L'agent est déconnecté du bus.